Logical Methods in Computer Science
Volume 21, Issue 2, 2025, pp. 8:1–8:95
https://lmcs.episciences.org/

Submitted Sep. 27, 2021
Published Apr. 24, 2025

A ROBUST GRAPH-BASED APPROACH TO OBSERVATIONAL EQUIVALENCE

DAN R. GHICA [a], KOKO MUROYA [b], AND TODD WAUGH AMBRIDGE [a]

[a] University of Birmingham, UK
e-mail address: d.r.ghica@cs.bham.ac.uk, t.waughambridge@bham.ac.uk

[b] National Institute of Informatics, Japan
e-mail address: kmuroya@nii.ac.jp

Abstract. We propose a new step-wise approach to proving observational equivalence, and in particular reasoning about fragility of observational equivalence. Our approach is based on what we call local reasoning. The local reasoning exploits the graphical concept of neighbourhood, and it extracts a new, formal, concept of robustness as a key sufficient condition of observational equivalence. Moreover, our proof methodology is capable of proving a generalised notion of observational equivalence. The generalised notion can be quantified over syntactically restricted contexts instead of all contexts, and also quantitatively constrained in terms of the number of reduction steps. The operational machinery we use is given by a hypergraph-rewriting abstract machine inspired by Girard's Geometry of Interaction. The behaviour of language features, including function abstraction and application, is provided by hypergraph-rewriting rules. We demonstrate our proof methodology using the call-by-value lambda-calculus equipped with (higher-order) state.

1. Introduction

1.1. **Context and motivation.** *Observational equivalence* [MJ69] is an old and central question in the study of programming languages. Two executable programs are observationally equivalent when they have the same behaviour. Observational equivalence between two program fragments (aka. terms) is the smallest congruence with respect to arbitrary program contexts. By formally establishing observational equivalence, one can justify compiler optimisation, and verify and validate programs.

There are two mathematical challenges in proving observational equivalence. Firstly, universal quantification over contexts is unwieldy. This has led to various indirect approaches to observational semantics. As an extremal case, *denotational semantics* provides a model-theoretic route to observational equivalence. There are also hybrid approaches that employ both denotational and operational techniques, such as *Kripke logical relations* [Sta85] and *trace semantics* [JR05]. Moreover, an operational and coinductive approach exists, under the name of *applicative bisimilarity* [Abr90].

The second challenge is *fragility* of observational equivalence. The richer a programming language is, the more discriminating power program contexts have and hence the less

DOI:10.46298/LMCS-21(2:8)2025

observational equivalences hold in the language. For example, the beta-law $(\lambda x.t)\ v \simeq t[v/x]$ is regarded as the fundamental observational equivalence in functional programming. However, it can be violated in the presence of a memory inspection feature like the one provided by the OCaml garbage-collection (Gc) module. A function that returns the size of a given program enables contexts to distinguish $(\lambda x.0)\ 100$ from 0, for example[1].

The fragility of observational equivalence extends to its proof methodologies. There have been studies of the impact that language features have on semantics and hence on proof methodologies for observational equivalence. The development of *game semantics* made it possible to give combinatorial, syntax-independent and orthogonal characterisations for classes of features such as state and control, e.g. the so-called "Abramsky cube" [Abr97, Ghi23], or to replace the syntactic notion of context by an abstracted *adversary* [GT12]. A classification [DNB12] and characterisation [DNRB10] of language features and their impact on reasoning has also been undertaken using logical relations. Applicative bisimilarity has been enriched to handle effects such as algebraic effects, local state, names and continuations [SKS07, KLS11, DGL17, SV18].

1.2. **Overview and contribution.** What is missing and desirable seems a general semantical framework with which one can directly analyse fragility, or robustness, of observational equivalences. To this end, we introduce a graphical abstract machine that implements *focussed hypernet rewriting*. We then propose a radically new approach to proving observational equivalence that is based on *step-wise* and *local* reasoning and centred around a concept of *robustness*. All these concepts will be rigorously defined in the paper.

The main contribution of the paper is rather conceptual, showing how the graphical concept of neighbourhood can be exploited to reason about observational equivalence, in a new and advantageous way. The technical development of the paper might seem quite elaborate, but this is because we construct a whole new methodology from scratch: namely, focussed hypernet rewriting and reasoning principles for it. These reasoning principles enable us to analyse fragility, or robustness, of observational equivalences in a formal way.

We introduce and use *hypernets* to represent (functional) programs with effects. Hypernets are an anonymised version of abstract syntax trees, where variables are simply represented as connections. Formally, hypernets are given by hierarchical hypergraphs. Hierarchy allows a hypergraph to be an edge label recursively. An extensive introduction to hypernets and rewriting of them can be found in the literature [GZ23].

Given a hypernet that represents a term, its evaluation is modelled by step-by-step traversal and update of the hypernet. Traversal steps implement depth-first search on the hypernet for a redex, and each update step triggers application of a rewrite rule to the hypernet. Traversal and update are interwoven strategically using a *focus* that is simply a dedicated edge passed around the hypernet. Importantly, updates are always triggered by a certain focus, and designed to only happen around the focus. We call this model of evaluation *focussed hypernet rewriting*.

There are mainly two differences compared with conventional reduction semantics. The first difference is the use of hypernets instead of terms. This makes renaming of variables irrelevant. The second difference is the use of the focus instead of evaluation contexts. In conventional reduction semantics, redexes are identified using evaluation contexts. Whenever

[1]See a concrete example written in OCaml, on the online platform *Try It Online*: `https://bit.ly/3TqnGOW`

the focus triggers an update of a hypernet, its position in the hypernet coincides with where the hole is in an evaluation context.

A new step-wise approach. This work takes a new coinductive, *step-wise*, approach to proving observational equivalence. We introduce a novel variant of the weak simulation dubbed *counting simulation*. We demonstrate that, to prove observational equivalence, it suffices to construct a counting simulation that is closed under contexts by definition. This approach is opposite to the known coinductive approach which uses applicative bisimilarity; one first constructs an applicative bisimulation and then proves that it is a congruence, typically using Howe's method [How96].

Local reasoning. In combination with our new step-wise approach, focussed hypernet rewriting facilitates what we call *local* reasoning. Our key observation is that, to obtain the counting simulation that is closed under contexts by definition, it suffices to simply trace sub-graphs and analyse their interaction with the focus. The interaction can namely be analysed by inspecting updates that happen around the focus and how these updates can interfere with the sub-graphs of interest. The reasoning principal here is the graphical concept of neighbourhood, or graph locality.

The local reasoning is a graph counterpart of analysing interaction between (sub-)terms and contexts using the conventional reduction semantics. In fact, it is not just a counterpart but an enhancement in two directions. Firstly, sub-graphs are more expressive than sub-terms; sub-graphs can represent parts of a program that are not necessarily well-formed. Secondly, the focus can indicate which part of a context is relevant in the interaction between the context and a term, which is not easy to make explicit in the conventional semantics that uses evaluation contexts.

Robustness. Finally, local reasoning extracts a formal concept of *robustness* in proving observational equivalence. Robustness is identified as the key sufficient condition that ensures two sub-graphs that we wish to equate interact with updates of a hypernet, which is triggered by the focus, in the *same* way; for example, if one sub-graph is duplicated (or discarded), the other is also duplicated (or discarded).

The concept of robustness helps us gain insights into fragility of observational equivalence. If robustness of two sub-graphs G, H fails, we obtain a counterexample, which is given by a rewrite rule that interferes with the two sub-graphs in different ways. Let G', H' be the two different results of interference (i.e. G' is the result of updating G, and H' is the result of updating H). There are two possibilities.

(1) The sub-graphs G, H are actually observationally equivalent. In this case, the counterexample suggests that the two different results G', H' should first be equated. The observational equivalence $G \simeq H$ we wish to establish is likely to depend on the ancillary observational equivalence $G' \simeq H'$.
(2) The observational equivalence $G \simeq H$ fails too. In this case, the counterexample provides the particular computation that violates the equivalence, in terms of a rewrite rule. We can conclude that the language feature that induces the computation violates the observational equivalence.

Generalised contextual equivalence. Using focussed hypernet rewriting, we propose a *generalised* notion of contextual equivalence. The notion has two parameters: a class of contexts and a preorder on natural numbers. The first parameter enables us to quantify over syntactically restricted contexts, instead of all contexts as in the standard notion. This can be used to identify a shape of contexts that respects or violates certain observational equivalences, given that not necessarily all arbitrarily generated contexts arise in program execution. The second parameter, a preorder on natural numbers, deals with numbers of steps it takes for program execution to terminate. Taking the universal relation recovers the standard notion of contextual equivalence. Another instance is contextual equivalence with respect to the greater-than-or-equal relation on natural numbers, which resembles the notion of *improvement* [San95, ADV20, ADV21] that is used to establish equivalence and also to compare efficiency of abstract machines. This instance of contextual equivalence is useful to establish that two programs have the same observable execution result, and also that one program terminates with fewer steps than the other.

1.3. **Organisation of the paper.** Section 2 provides a gentle introduction to our graph-based approach to modelling program evaluation, and reasoning about observational equivalence with the key concepts of locality and robustness. Section 3 formalises the graphs we use, namely hypernets. The rest of the paper is in two halves.

In the first half, we develop our reasoning framework, targeting the linear lambda-calculus. Although linear lambda-terms have restricted expressive power, they are simple enough to demonstrate the development throughout. Section 4 presents the hypernet representation of linear lambda-terms. Section 5 then presents our operational semantics, i.e. focussed hypernet rewriting. Section 6 formalises it as an abstract machine called *universal abstract machine (UAM)*.

Section 7 sets the target of our proof methodology, introducing the generalised notion of contextual equivalence. Section 8 presents our main technical contributions: it formalises the concept of robustness, and presents our main technical result which is the sufficiency-of-robustness theorem (Theorem 8.11).

In the second half, we extend our approach to the general (non-linear) lambda-calculus equipped with store. Section 9 describes how the hypernet representation can be adapted. Section 10 shows how the UAM can be extended accordingly, and presents the *copying* UAM. Section 11 formalises observational equivalence between lambda-terms by means of contextual equivalence between hypernets. Section 12 then demonstrates our approach by proving some example equivalences for the call-by-value lambda-calculus extended with state. The choice of the language here is pedagogical; our methodology can accommodate other effects as long as they are deterministic.

Finally, Section 13 discusses related and future work, concluding the paper. Some details of proofs are presented in Appendix.

2. A gentle introduction

2.1. **Hypernets.** Compilers and interpreters deal with programs mainly in the form of an abstract syntax tree (AST) rather than text. It is broadly accepted that such a data structure is easier to manipulate algorithmically. Somewhat curiously perhaps, reduction semantics (or small-step operational semantics), which is essentially a list of rules for program

manipulation, is expressed using text rather than the tree form. In contrast, our graph-based semantics is expressed as algorithmic manipulations of the data structure that represents syntax.

Let us demonstrate our graphical representation, using the beta-law

$$\lambda y.(\lambda x.(\lambda y.y\ x)\ (\lambda z.z))\ (\lambda x.x\ y) \simeq \lambda y.(\lambda w.w\ (\lambda x.x\ y))\ (\lambda z.z) \tag{2.1}$$

in the linear lambda-calculus. We use colours to clarify some variable scopes. This law substitutes $\lambda x.x\ y$ for the variable x, and in doing so, the bound variable y has to be renamed to w so it does not capture the variable y in $\lambda x.x\ y$.

Our first observation is that ASTs are not satisfactory to represent syntax, when it comes to define operational semantics. They contain more syntactic details than necessary, namely by representing variables using names. This makes an operation on terms like substitution a global affair. To avoid variable capturing, substitution needs to clarify the scope of each variable and appropriately rename some variables.

Figure 1a shows the beta-law (2.1) using ASTs. The scope of each variable is not obvious in the ASTs, which is why we keep using colours to distinguish variable scopes. The law deletes the four red nodes of the left AST, and connects two red arrows to represent substitution for x. Additionally, all occurrences of the variable y has to be renamed to w.

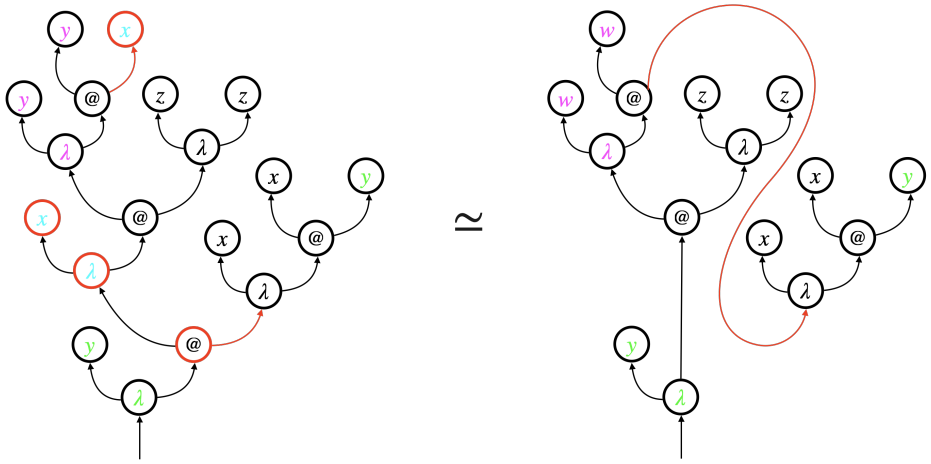
We propose hypernets as an alternative graph representation. Hypernets, inspired by *proof nets* [Gir87], replace variable names with virtual connection, and hence keep variables anonymous. Binding structures and scopes are made explicit by (dashed) boxes around sub-graphs.

Figure 1b shows the same beta-law (2.1) using hypernets instead of ASTs. Each bound variable is simply represented by an arrow that points at the left edge of the associated dashed box. For example, the upper one of the two red arrows in the left hypernet represents the bound variable x. It points at the left edge of the red dashed box that represents the scope of the variable. The dashed box is connected to the corresponding binder (λ).

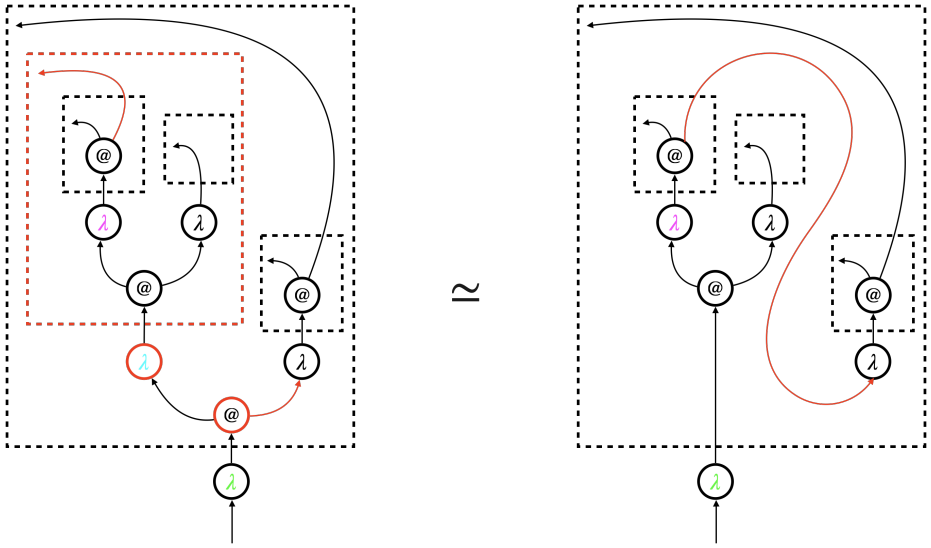
The beta-law requires relatively local changes to hypernets. In Figure 1b, the two red nodes are deleted, the associated red dashed box is also deleted, and the two red arrows are connected to represent substitution for x. There is no need for renaming y, as it is simply represented by an anonymous arrow.

Remark 2.1 (Arrows representing bound variables)**.** In hypernets, the arrow representing a bound variable points at the left edge of the associated dashed box. In other graphical notations (e.g. proof nets [Gir87]), the bound variable would be connected to the corresponding binder (λ), as shown by red arrows in Figure 1c. We treat these red arrows as mere *decorations*, and exclude them from the formalisation of hypernets. We find that boxes suffice to delimit the scope of variables and sub-terms. Exclusion of decorations also simplifies the formalisation by reducing the number of loops in each hypernet. ■

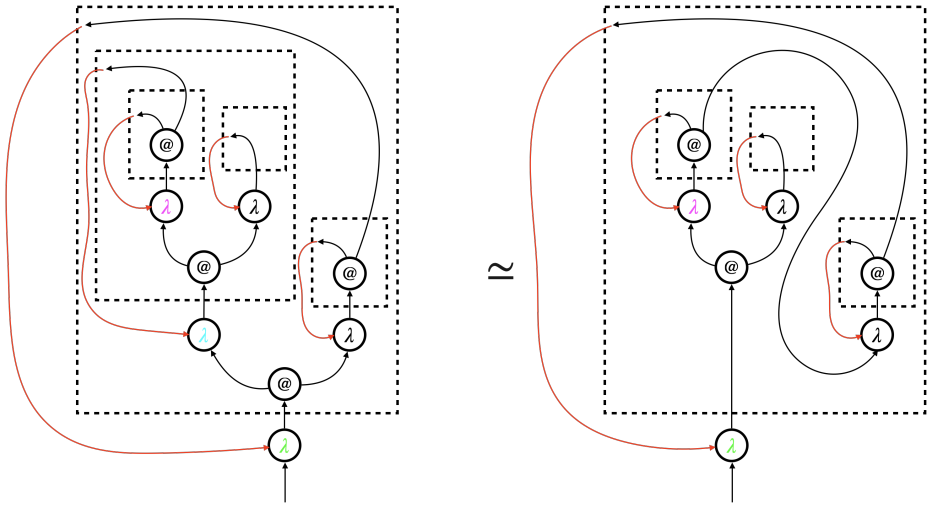
2.2. **Focussed hypernet rewriting.** The main difference between a *law* (an equation) and a *reduction* is that the former can be applied in any context, at any time, whereas the latter must be applied *strategically*, in a particular (evaluation) context and in a particular order. Different reduction strategies, for instance, make different programming languages out of the same calculus.



(a) ASTs



(b) Hypernets



(c) Decorated hypernets

Figure 1: The beta-law $\lambda y.(\lambda x.(\lambda y.y\ x)\ (\lambda z.z))\ (\lambda x.x\ y) \simeq \lambda y.(\lambda w.w\ (\lambda x.x\ y))\ (\lambda z.z)$

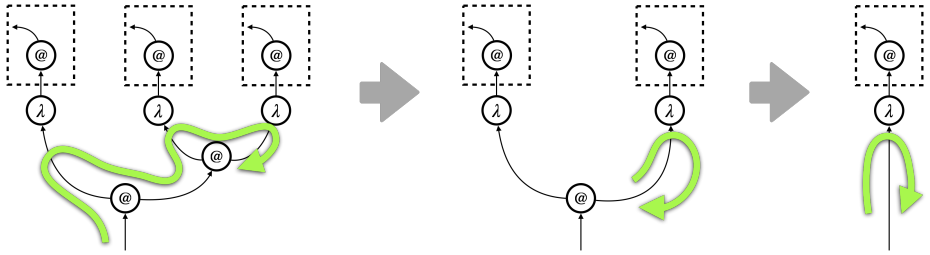


Figure 2: Reduction as graph traversal and update

The question to be addressed here is how to define strategies for determining redexes in hypernets. Our operational semantics, i.e. *focussed hypernet rewriting*, combines graph traversal with update, and exploits the traversal to search for a redex.

Let us illustrate focussed hypernet rewriting, using the call-by-value reduction of the linear lambda-term $(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.z))$ as shown in Figure 2. The thicker green arrows are not part of the hypernets but they show the traversal. The reduction proceeds as follows.

(1) The depth-first traversal witnesses that the abstraction $\lambda x.x$ is a value, and that the sub-term $(\lambda y.y)\ (\lambda z.z)$ contains two abstractions and it is ready for the beta-reduction. In the reduction, an application node (@) and its matching abstraction node (λ) are deleted, and the associated dashed box is removed. The argument $\lambda z.z$ is then connected to the bound variable y, yielding the second hypernet.
(2) The traversal continues on the resultant hypernet (representing $(\lambda x.x)\ (\lambda z.z)$), confirming that the result $\lambda z.z$ of the beta-reduction is a value. Note that the abstraction $\lambda x.x$ has already been inspected in the previous step, so the traversal does not repeat the inspection. It only witnesses the abstraction $\lambda z.z$ at this stage. The beta-reduction is then triggered, yielding the third and final hypernet representing $\lambda z.z$.
(3) The traversal confirms that the result $\lambda z.z$ of the beta-reduction is a value, and it finishes.

We implement focussed hypernet rewriting, in particular the graph traversal (the thick green arrows in Figure 2), using a dedicated node dubbed *focus*. A focus can be in three modes: searching (?), backtracking (✓), and triggering (↯). The first two modes implement the depth-first traversal, and the last mode triggers update of the underlying hypernet. Figure 3 shows how focussed hypernet rewriting actually proceeds[2], given the linear term $(\lambda x.x)\ ((\lambda y.y)\ (\lambda z.z))$. The black nodes are the focus. The first eight steps $\rightsquigarrow$ altogether implement the thick green arrow in the first part of Figure 2. At the end of these steps, the focus changes to ↯, signalling that the hypernet is ready for the beta-reduction. What follows is an update of the hypernet, which resets the focus to the searching mode (?), so the traversal continues and triggers further update.

Evaluation of a program P starts, when the ?-focus enters the hypernet representing P from the bottom. Evaluation successfully finishes, when the ✓-focus exits a hypernet from the bottom.

We will formalise focussed hypernet rewriting as an abstract machine (see [Pit00] for a comprehensive introduction). The machine has two kinds of transitions: one for the traversal, and the other for the update. It is important that the focus governs transitions; a traversal

[2]An interpreter and visualiser can be accessed online at `https://tnttodda.github.io/Spartan-Visualiser/`

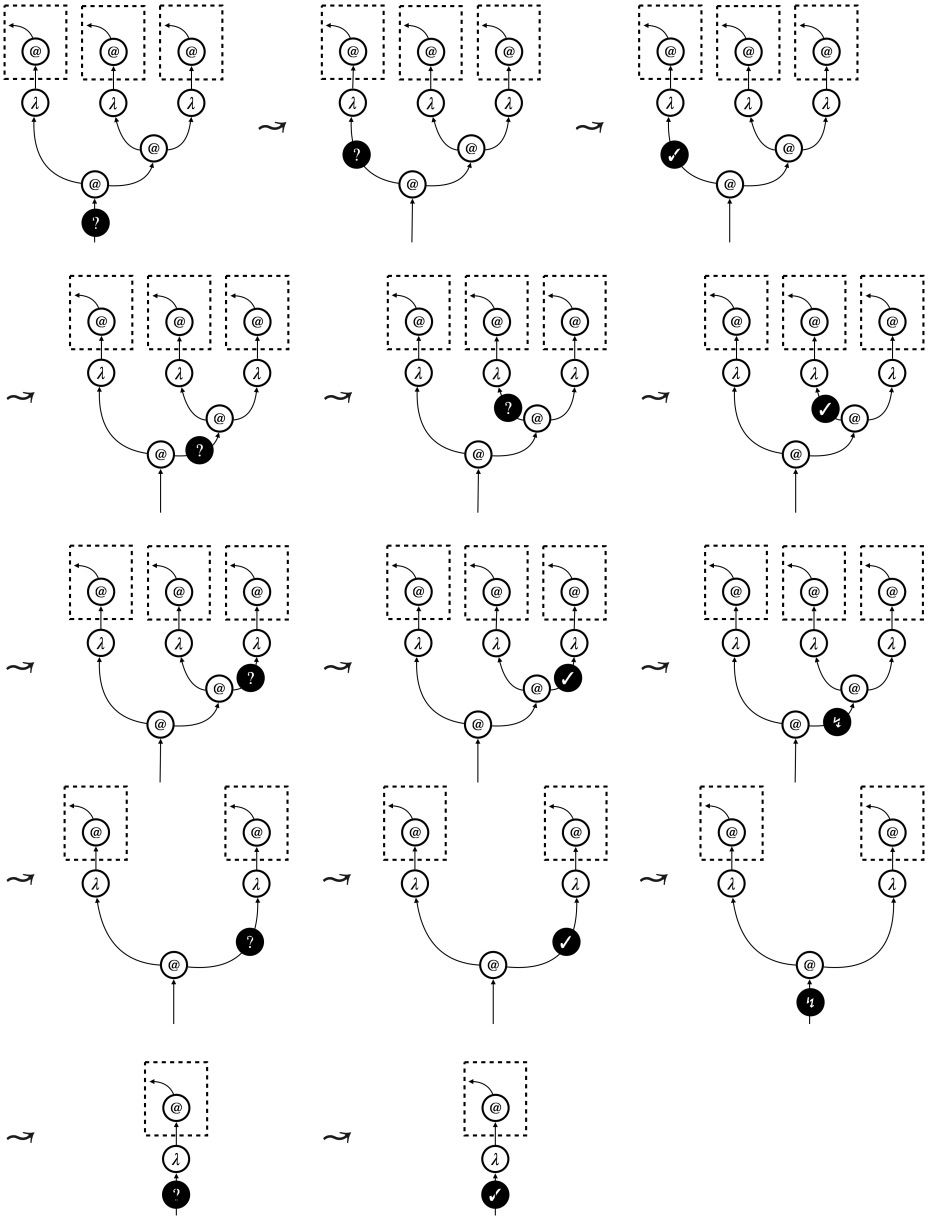


Figure 3: Graph traversal and update with a focus

transition or an update transition is selected according to the mode of the focus. It is the focus that implements the traversal, triggers the update, and hence realises the call-by-value reduction strategy.

2.3. **Step-wise local reasoning, and robustness.** Finally we overview the reasoning principle that focussed hypernet rewriting enables, which leads to our main theorem, sufficiency-of-robustness theorem (Theorem 8.11).

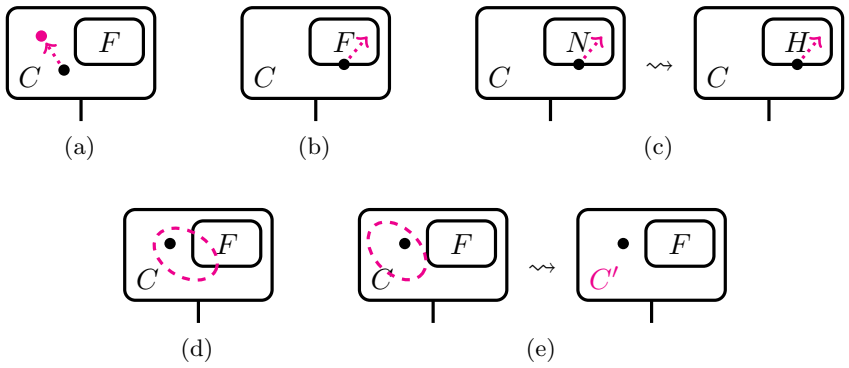


Figure 4: Example scenarios of case analysis for (2.2), where $F \in \{N, H\}$

Using focussed hypernet semantics, this work takes a new coinductive, step-wise, approach to proving observational equivalence. We will introduce a new variant of weak simulation dubbed *counting simulation*. A counting simulation is a relation on *focussed* hypernets that are hypernets with a focus. We write $\dot{N}$ to indicate that a hypernet N contains a focus.

Our proof of an observational refinement $N \preceq H$, which is the asymmetric version of observational equivalence $N \simeq H$, proceeds as follows.

(1) We start with the relation $\lhd := \{(N, H)\}$ on hypernets. We call it *pre-template*.
(2) We take the *contextual closure* $\overline{\lhd}$ of the pre-template $\lhd$. It is defined by $\dot{C}[N, \ldots, N] \mathrel{\overline{\lhd}} \dot{C}[H, \ldots, H]$ for an arbitrary focussed (multi-hole) context $\dot{C}$.
(3) We show that $\overline{\lhd}$ is a counting simulation.

Once we establish the counting simulation $\overline{\lhd}$, soundness of counting simulation asserts that the pre-template $N \lhd H$ implies observational refinement $N \preceq H$.

The key part of the observational refinement proof is therefore showing that $\overline{\lhd}$ is a counting simulation. Put simply, this amounts to show the following: for any $\dot{C}[N, \ldots, N] \mathrel{\overline{\lhd}} \dot{C}[H, \ldots, H]$ and a transition $\dot{C}[N, \ldots, N] \to \dot{P}$, there exists a focussed context $\dot{C}'$ that satisfies the following.

$$\begin{array}{ccccc} \dot{C}[N,\ldots,N] & \longrightarrow & \dot{P} & \longrightarrow^{*} & \dot{C}'[N,\ldots,N] \\ \overline{\lhd} \,\vdots & & & & \vdots\, \overline{\lhd} \\ \dot{C}[H,\ldots,H] & & \longrightarrow^{*} & & \dot{C}'[H,\ldots,H] \end{array} \tag{2.2}$$

Above, black parts are universally quantified, and magenta parts are existentially quantified. The arrow $\to^*$ represents an arbitrary number of transitions $\to$. This situation (2.2) asserts that, after a few transitions from $\dot{P}$ and $\dot{C}[H, \ldots, H]$, we can obtain two focussed hypernets that can be decomposed using the new context $\dot{C}'$ and the sub-graphs N, H.

Our important observation is that (2.2) can be established by elementary case analysis of interaction between the sub-graphs N, H and what happens around the focus in $\dot{C}[N, \ldots, N] \to \dot{P}$. This is because updates of a hypernet always happen around the $\lightning$-focus, and the ?-focus and the ✓-focus (representing the graph traversal) move according to its neighbourhood. The analysis is hence centred around the graphical concept of neighbourhood, or graph locality. There are three possible cases of the interaction.

Case (i) Move inside the context: The ?-focus or the ✓-focus, which implements the depth-first traversal, simply moves inside the context C (see Figure 4a). Because any move of the focus is only according to its neighbourhood, the move solely depends on the context C. In other words, the sub-graphs N, H have no interaction with the focus. In this case, we can conclude that we are always in (2.2).

Case (ii) Visit to the sub-graphs: The ?-focus visits the sub-graphs N, H (see Figure 4b). This is the case where the ?-focus actually interacts with N, H; what happens after entering of the focus depends on N, H. We identify a sufficient condition of the pre-template $\lhd$, dubbed *safety*, for (2.2) to hold.

A typical example of safe pre-templates is the pre-templates that are induced by rewrite rules of hypernets. Figure 4c illustrates what happens to such a pre-template. The visit of the ?-focus to N triggers the rewrite rule, and actually turns N into H.

Note that the case where the ✓-focus visits N, H boils down to the visit of the ?-focus instead, because the ✓-focus implements backtracking of graph traversal.

Case (iii) Update of the hypernets: The $\lightning$-focus triggers a rewrite rule and updates the hypernet (see Figure 4d). This is the case where the $\lightning$-focus interacts with N, H; the update may involve N, H in a non-trivial manner. We identify a sufficient condition of the pre-template $\lhd$, relative to the triggered rewrite rule, dubbed *robustness*, for (2.2) to hold.

An example scenario of robustness is where the update only affects parts of the context C; see Figure 4e. In this scenario, the $\lightning$-focus does not really interact with N, H. The sub-graphs N, H are preserved, and we can take the new context C' with the same number of holes as C that makes (2.2) hold.

Another example scenario of robustness is where the update duplicates (or eliminates) N, H without breaking them. We can take the context C' that has more (or less) holes to make (2.2) hold. ■

The above case analysis reveals sufficient conditions, namely safety and robustness, to make (2.2) hold and hence make the context closure $\overline{\lhd}$ a counting simulation. Combining this with soundness of counting simulation, we obtain our main theorem, sufficiency-of-robustness theorem (Theorem 8.11). It can be informally stated as follows.

Theorem (Sufficiency-of-robustness theorem (Theorem 8.11), informally)**.** *A robust and safe pre-template $N \lhd H$ induces observational refinement $N \preceq H$.*

3. Preliminaries: Hypernets

We start formalising the ideas described in the previous section, by first defining hypernets. We opt for formalising hypernets as hypergraphs, following the literature [BGK$^+$22a, BGK$^+$22b, BGK$^+$22c, AGSZ22] on categorically formalising string diagram rewriting using hypergraphs.

Let $\mathbb{N}$ be the set of natural numbers. Given a set X we write by X^* the set of elements of the free monoid over X. Given a function $f : X \to Y$ we write $f^* : X^* \to Y^*$ for the pointwise application (map) of f to the elements of X^*.

3.1. **Monoidal hypergraphs and hypernets.** Hypernets have a couple of distinctive features in comparison with ordinary graphs. The first feature of hypernets is that they have "dangling edges" (see Figure 1b); a hypernet has one incoming arrow with no source, and it may have outgoing arrows with no targets. To model this, we use *hypergraphs*—we formalise what we have been calling edges (i.e. arrows) as *vertices* and what we have been calling nodes (i.e. circled objects) as *hyperedges* (i.e. edges with arbitrary numbers of sources and targets). More specifically, we use what we call *interfaced labelled monoidal hypergraphs* that satisfies the following.

(0) Each arrow (modelled as a vertex) and each circled object (modelled as a hyperedge) are labelled.
(1) Each circled object (modelled as a hyperedge) is adjacent to distinct arrows (modelled as vertices).
(2) Each arrow (modelled as a vertex) is adjacent to at most two circled objects (modelled as hyperedges).
(3) The label of a circled object (modelled as a hyperedge) is always consistent with the number and labelling of its endpoints.
(4) Dangling arrows are ordered, and each arrow has at least a source or a target.

Definition 3.1 (Monoidal hypergraphs)**.** A *monoidal hypergraph* is a pair (V, E) of finite sets, *vertices* and *(hyper)edges* along with a pair of functions $S : E \to V^*$, $T : E \to V^*$ defining the *source list* and *target list*, respectively, of an edge.

Definition 3.2 (Interfaced labelled monoidal hypergraphs)**.** An *interfaced labelled monoidal hypergraph* consists of a monoidal hypergraph, a set of vertex labels L_V, a set of edge labels L_E, and labelling functions $f_V \colon V \to L_V$, $f_E \colon E \to L_E$ such that:

(1) For any edge $e \in E$, its source list $S(e)$ consists of distinct vertices, and its target list $T(e)$ also consists of distinct vertices.
(2) For any vertex $v \in V$ there exists at most one edge $e \in E$ such that $v \in S(e)$ and at most one edge $e' \in E$ such that $v \in T(e')$.
(3) For any edges $e_1, e_2 \in E$ if $f_E(e_1) = f_E(e_2)$ then $f_V^*\big(S(e_1)\big) = f_V^*\big(S(e_2)\big)$, and $f_V^*\big(T(e_1)\big) = f_V^*\big(T(e_2)\big)$.
(4) If a vertex belongs to the target (resp. source) list of no edge we call it an *input* (resp. *output*). Inputs and outputs are respectively ordered, and no vertex is both an input and an output.

Notation 3.3 (Types of circled objects (i.e. hyperedges))**.** Definition 3.2 (3) makes it possible to use labels of arrows (i.e. vertices) as *types* for labels of circled objects (i.e. hyperedges). For each $m \in L_E$, we can associate it a type and write $m \colon x \Rightarrow x'$, where $x, x' \in L_V^*$ satisfy $x = f_V^*\big(S(e)\big)$ and $x' = f_V^*\big(T(e)\big)$ for any $e \in E$ such that $f_E(e) = m$.

Notation 3.4 (Types of interfaced labelled monoidal hypergraphs)**.** The concept of type can be extended to a whole interfaced labelled monoidal hypergraph G. Let I, O be the lists of inputs and outputs, respectively, of G. We associate G a type and write $G \colon f_V^*(I) \Rightarrow f_V^*(O)$. In the syntax for lists of inputs and outputs we use $\otimes$ to denote concatenation and define ϵ to be the empty list, and $A^{\otimes 0} := \epsilon$, $A^{\otimes(n+1)} := A \otimes A^{\otimes n}$ for any label A and $n \in \mathbb{N}$.

In the sequel, when we say hypergraphs we always mean interfaced labelled monoidal hypergraphs.

We sometimes permute inputs and outputs of a hypergraph. Such permutation yields another hypergraph.

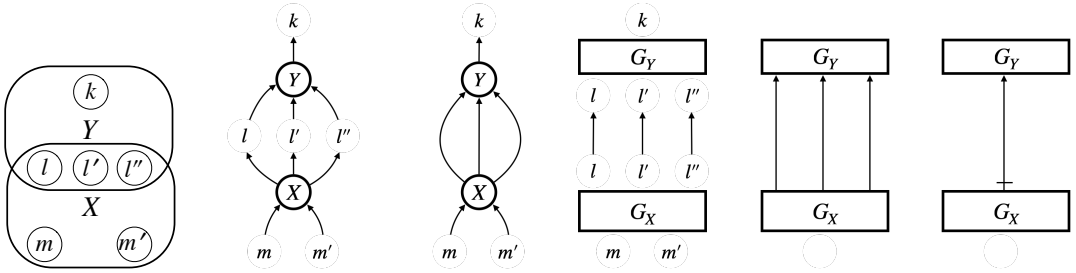


Figure 5: The hypergraph G_{ex}

Definition 3.5 (Interface permutation)**.** Let G be a hypergraph with an input list $i_1, \ldots, i_n$ and an output list $o_1, \ldots, o_m$. Given two bijections ρ and ρ' on sets $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$, respectively, we write $\Pi_\rho^{\rho'}(G)$ to denote the hypergraph that is defined by the same data as G except for the input list $i_{\rho(1)}, \ldots, i_{\rho(n)}$ and the output list $o_{\rho'(1)}, \ldots, o_{\rho'(m)}$.

The second distinctive feature of hypernets is that they have dashed boxes that indicate the scope of variable bindings (see Figure 1b). We formalise these dashed boxes by introducing *hierarchy* to hypergraphs. The hierarchy is implemented by allowing a hypergraph to be the label of a hyperedge. As a result, informally, hypernets are nested hypergraphs, up to some finite depth, using the same sets of labels. We here present a relatively intuitive definition of hypernets; Appendix A discusses an alternative definition of hypernets[3].

Definition 3.6 (Hypernets)**.** Given a set of vertex labels L and edge labels M we write $\mathcal{H}(L, M)$ for the set of hypergraphs with these labels; we also call these *level-0 hypernets* $\mathcal{H}_0(L, M)$. We call *level-*$(k{+}1)$ *hypernets* the set of hypergraphs

$$\mathcal{H}_{k+1}(L, M) = \mathcal{H}\Big(L, M \cup \bigcup_{i \leq k} \mathcal{H}_i(L, M)\Big).$$

We call *hypernets* the set $\mathcal{H}_\omega(L, M) = \bigcup_{i \in \mathbb{N}} \mathcal{H}_i(L, M)$.

Terminology 3.7 (Boxes and depth)**.** An edge labelled with a hypergraph is called *box* edge, and a hypergraph labelling a box edge is called *content*. Edges of a hypernet G are said to be *shallow*. Edges of nesting hypernets of G, i.e. edges of hypernets that recursively appear as edge labels, are said to be *deep* edges of G. Shallow edges and deep edges of a hypernet are altogether referred to as edges *at any depth*.

[3]Another, slightly different, definition of hypernets is given in [AGSZ23, Section 4]. The difference is motivated by desired support of categorical graph rewriting, which requires certain properties to hold. These properties are sensitive to the definition.

3.2. **Graphical conventions.** A hypergraph G_{ex} with vertices $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$ and edges $E = \{e_1, e_2\}$ such that

$$\begin{aligned} S(e_0) &= \{v_0, v_1\} \\ T(e_0) = S(e_1) &= \{v_2, v_3, v_4\} \\ T(e_1) &= \{v_5\} \\ f_V &= \{v_0 \mapsto m, v_1 \mapsto m', v_2 \mapsto l, v_3 \mapsto l', v_4 \mapsto l'', v_5 \mapsto k\} \\ f_E &= \{e_0 \mapsto X, e_1 \mapsto Y\} \end{aligned}$$

is normally represented as Figure 5a. However, we find this style of representing hypergraphs awkward for understanding their structure. We will often graphically represent hypergraphs as graphs, as in Figure 5b, by (i) marking vertices with their labels and mark hyperedges with their labels circled, and (ii) connecting input vertices and output vertices with a hyperedge using arrows.

Recall that node labels are often determined in hypergraphs, thanks to typing, e.g. $X \colon m \otimes m' \Rightarrow l \otimes l' \otimes l''$. We accordingly omit node labels to avoid clutter, as in Figure 5c, letting arrows connect circles directly.

Sometimes we draw a hypergraph by connecting its sub-graphs using extra arrows. Sub-graphs are depicted as boxes. For the hypergraph G_{ex}, we can think of the sub-graph $G_X = (\{v_0, v_1, v_2, v_3, v_4\}, \{e_0\})$ and the sub-graph $G_Y = (\{v_2, v_3, v_4, v_5\}, \{e_1\})$. We may draw G_{ex} in three ways, as in Figure 5d–5f:

- In Figure 5d, we use extra arrows connecting node labels l, l', l'' directly, with intention that two occurrences of l (or, l', l'') are graphical representations of the same node v_2 (or v_3, v_4).
- In Figure 5e, we omit node labels entirely, assuming that they are obvious from context.
- In Figure 5f, we further replace the three extra arrows with a single arrow, given that the entire output type of $G_X \colon m \otimes m' \Rightarrow l \otimes l' \otimes l''$ matches the input type of $G_Y \colon l \otimes l' \otimes l'' \Rightarrow k$. The single arrow comes with a dash across, which indicates that the arrow represents a bunch of parallel arrows.

The final convention is about box edges; a box edge (i.e. an edge labelled by a hypernet) is depicted by a dashed box decorated with its content (i.e. the labelling hypernet).

4. Representation of the call-by-value untyped linear lambda-calculus

In this section we introduce specific label sets that we use to represent lambda-terms as hypernets. We begin with the untyped linear lambda-calculus extended with arithmetic. This is a fairly limited language in terms of expressive power. Although simple, it is interesting enough to demonstrate our reasoning framework. We present a translation $(-)^\dagger$ of linear lambda-terms in this section, and will adapt it to the general lambda-terms in Section 9.

Lambda-terms are defined by the BNF $t ::= x \mid \lambda x.t \mid t\ t \mid n \mid t\ op\ u$, where $n \in \mathbb{N}$ and $op \in \{+, -\}$. We assume alpha-equivalence on terms, and assume that bound variables are distinct in a term. A term t is *linear* when each variable appears exactly once in t.

First, recall that each edge label $m \in M$ of a hypernet $G \in \mathcal{H}_\omega(L, M)$ comes with a *type* $m \colon X \Rightarrow X'$ where $X, X' \in L^*$. Even though lambda-terms are untyped here, we use edges to represent term constructors, and use edge types (i.e. node labels) to distinguish thunks from terms. Namely, the term type is denoted by $\star$, and a thunk type with n bound variables is denoted by $T^n(\star)$.

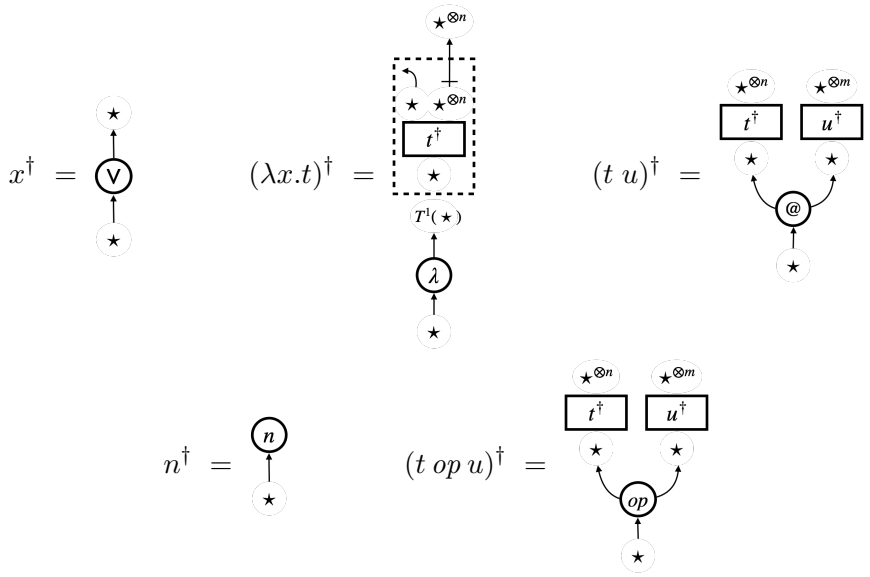


Figure 6: Inductive translation $(-)^\dagger$ of linear lambda-terms

Figure 6 shows inductive translation of linear lambda-terms to hypernets $\mathcal{H}_\omega(L_{\text{lin}}, M_{\text{lin}})$ where

$$L_{\text{lin}} = \{\star\} \cup \{T^n(\star) \mid n \in \mathbb{N}\}, \tag{4.1}$$

$$M_{\text{lin}} = \{\mathsf{V}\colon \star \Rightarrow \star,\ \lambda\colon \star \Rightarrow T^1(\star),\ \vec{@}\colon \star \Rightarrow \star^{\otimes 2},\ +\colon \star \Rightarrow \star^{\otimes 2},\ -\colon \star \Rightarrow \star^{\otimes 2}\} \cup \{n\colon \star \Rightarrow \epsilon \mid n \in \mathbb{N}\}. \tag{4.2}$$

In general, a term t with n free variables is translated into $t^\dagger\colon \star \Rightarrow \star^{\otimes n}$. Each term constructor is turned into an edge as follows.

- Any variable becomes an anonymous edge $\mathsf{V}\colon \star \Rightarrow \star$. We represented a variable as a single arrow in Section 2 (see e.g. Figure 1b and Figure 3), but this means a variable would become an empty graph (i.e. a hypernet with no edge). We rather use the anonymous edge $\mathsf{V}\colon \star \Rightarrow \star$ to prevent an empty graph from labelling a box edge and hence being a box content. This is for technical reasons as they simplify our development.
- Abstraction becomes $\lambda\colon \star \Rightarrow T^1(\star)$; it constructs a term, taking one thunk that has one bound variable. A thunk that has one bound variable and n free variables is represented by a box edge of type $T^1(\star) \Rightarrow \star^{\otimes n}$ whose content is $t^\dagger\colon \star \Rightarrow \star \otimes \star^{\otimes n}$. Note that the box has one less output than its content. We emphasise this graphically, by bending the arrow that is connected to the leftmost $\star$ of the type $\star \otimes \star^{\otimes n}$.
- Application becomes $\vec{@}\colon \star \Rightarrow \star^{\otimes 2}$; it constructs a term, taking two terms as arguments. This translation is for the call-by-value evaluation strategy; both of the arguments are not thunks, and hence they will be evaluated before the application is computed.
- Each natural number $n \in \mathbb{N}$ becomes $n\colon \star \Rightarrow \epsilon$; it is a term, taking no arguments.
- Arithmetic operations becomes $+\colon \star \Rightarrow \star^{\otimes 2}$ and $-\colon \star \Rightarrow \star^{\otimes 2}$; they construct a term, taking two terms as arguments. The translation for these operations has the same shape as that for application.

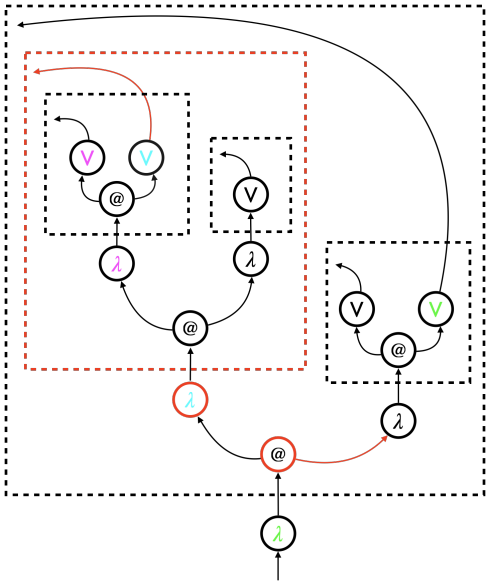


Figure 7: The hypernet $\big(\lambda y.(\lambda x.(\lambda y.y\ x)\ (\lambda z.z))\ (\lambda x.x\ y)\big)^{\dagger} : \star \Rightarrow \epsilon$

Figure 7 shows an example of the translation. Note that this is different from the left hand side of Figure 1b; each variable is now represented by the anonymous V-edge.

5. Focussed hypernet rewriting—the Universal Abstract Machine

In this section we present focussed hypernet rewriting. It will be formalised as an abstract machine dubbed *universal abstract machine (UAM)*. This abstract machine is "universal" in a sense of the word similar to the way it is used in "universal algebra" rather than in "universal Turing machine". It is a general abstract framework in which a very wide range of concrete abstract machines can be instantiated by providing *operations* and their *behaviour*.

5.1. Operations and focus.

The first parameter of the UAM is given by a set of *operations* $\mathbb{O} = \mathbb{O}_{\checkmark} \uplus \mathbb{O}_{\lightning}$. Operations are classified into two: *passive* operations $\mathbb{O}_{\checkmark}$ that construct evaluation results (i.e. values) and *active* operations $\mathbb{O}_{\lightning}$ that realise computation. We let $\phi, \phi_{\checkmark}, \phi_{\lightning}$ range over $\mathbb{O}, \mathbb{O}_{\checkmark}, \mathbb{O}_{\lightning}$ respectively. The edge labels in L_{lin} (4.2), except for V, are examples of operations:

passive operations	active operations
$\lambda : \star \Rightarrow T^1(\star)$	$\overrightarrow{@} : \star \Rightarrow \star^{\otimes 2}$
$n : \star \Rightarrow \epsilon$ for each $n \in \mathbb{N}$	$op : \star \Rightarrow \star^{\otimes 2}$ where $op \in \{+, -\}$

In general, each operation $\phi \in \mathbb{O}$ has a type $\star \Rightarrow \star^{\otimes m} \otimes \bigotimes_{i=1}^{k} T^{n_i}(\star)$ where $m, k, n_1, \ldots, n_k \in \mathbb{N}$. This means that each operation takes m arguments and k thunks, and the i-th thunk has n_i bound variables.

Given the operation set $\mathbb{O}$, the UAM acts on hypernets $\mathcal{H}_{\omega}(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O}))$ where

$$M_{\mathrm{lin}}(\mathbb{O}) = \mathbb{O} \cup \{\mathsf{V} : \star \Rightarrow \star\} \cup \{? : \star \Rightarrow \star,\ \checkmark : \star \Rightarrow \star,\ \lightning : \star \Rightarrow \star\}. \tag{5.1}$$

transitions		focus	provenance
search transitions		?, ✓	intrinsic
rewrite transitions	substitution transitions	↯	
	behaviour $B_{\mathbb{O}}$ (compute transitions)		extrinsic

Table 1: Transitions of the UAM

We let ℓ range over L_{lin}. For box edges, we impose the following type discipline: each box edge must have a type $T^n(\star) \Rightarrow \star^{\otimes m}$ with its content having a type $\star \Rightarrow \star^{\otimes(n+m)}$.

The last three elements of (5.1) are *focuses*. In the UAM, focuses are edges with the dedicated labels $?, \checkmark, \lightning$ of type $\star \Rightarrow \star$. Each label represents one of the three modes of the focus:

label	mode
?	searching
✓	backtracking
↯	triggering an update

As illustrated in Section 2.2, focussed hypernet rewriting implements program evaluation by combining (i) depth-first graph traversal and (ii) update (rewrite) of a hypernet. Focuses are the key element of this combination; they determine which action (i.e. traversal or rewrite) to be taken next, and they indicate where a redex of the rewrite is.

We refer to a hypernet that contains one focus as *focussed* hypernet. Given a focussed hypernet, we refer to the hypernet without the focus as *underlying* hypernet.

5.2. **Transitions—overview.** Table 1 summarises classifications of transitions of the UAM.

The first classification is according to the focuses: *search* transitions for the ?-focus and the ✓-focus, implementing the depth-first search of redexes, and *rewrite* transitions for the ↯-focus, implementing rewrite of the underlying hypernet. Rewrite transitions are further classified into two: *substitution* transitions for edges labelled by V, and *behaviour* $B_{\mathbb{O}}$ for (active) operations.

The next classification is according to provenance: *intrinsic* transitions that are inherent to the UAM, and *extrinsic* transitions that are not. While search transitions and substitution transitions constitute intrinsic transitions, the behaviour $B_{\mathbb{O}}$ solely provides extrinsic transitions. In fact, the behaviour $B_{\mathbb{O}}$ is the second parameter of the UAM.

Figure 8 shows an example of the UAM execution. This evaluates the linear-term $(\lambda x.x)\ 1$, and the execution starts with the underlying hypernet $((\lambda x.x)\ 1)^{\dagger}$. It demonstrates all the three kinds of transitions: search transitions, substitution transitions, and the behaviour of application $(\overrightarrow{@})$.

5.3. **Intrinsic transitions.** Search transitions are possible for the ?-focus and the ✓-focus, and they implement the depth-first search of redexes. They are specified by *interaction rules* depicted in Figure 9a–9e. The ?-focus interacts with what is connected above, and the ✓-focus interacts with what is connected below. From the perspective of program evaluation, the interaction rules specify the left-to-right call-by-value evaluation of arguments.

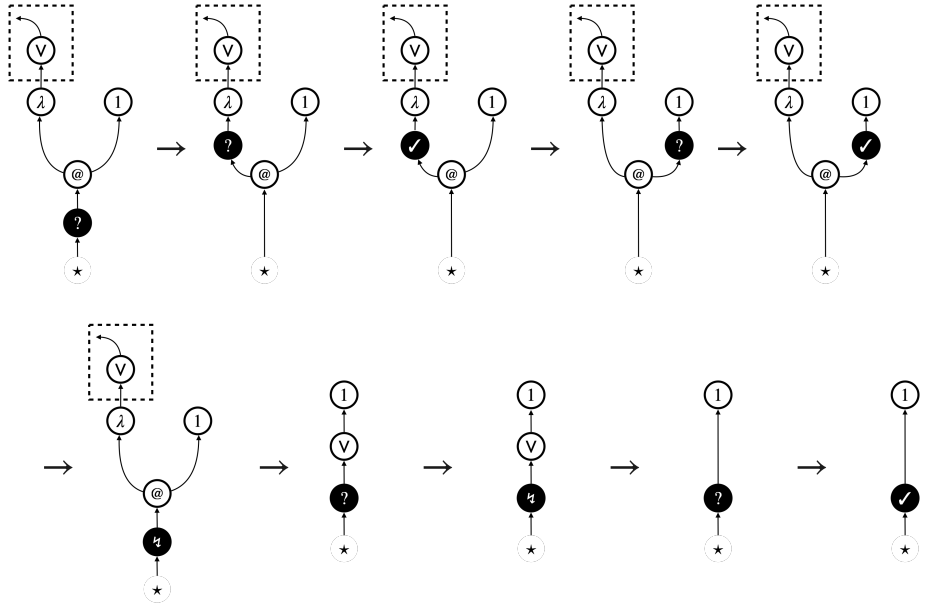


Figure 8: An example of the UAM execution on $((\lambda x.x)\ 1)^{\dagger}$

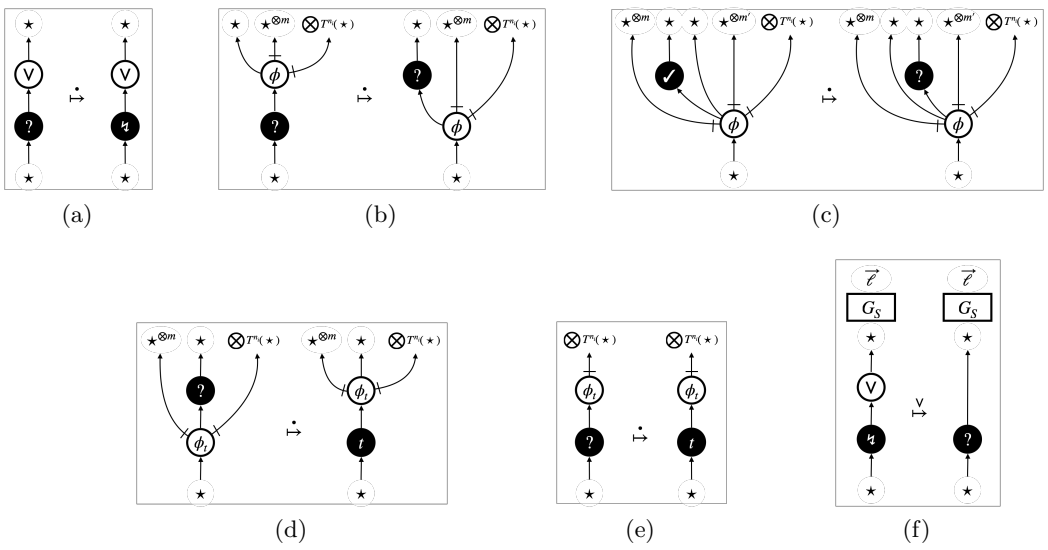
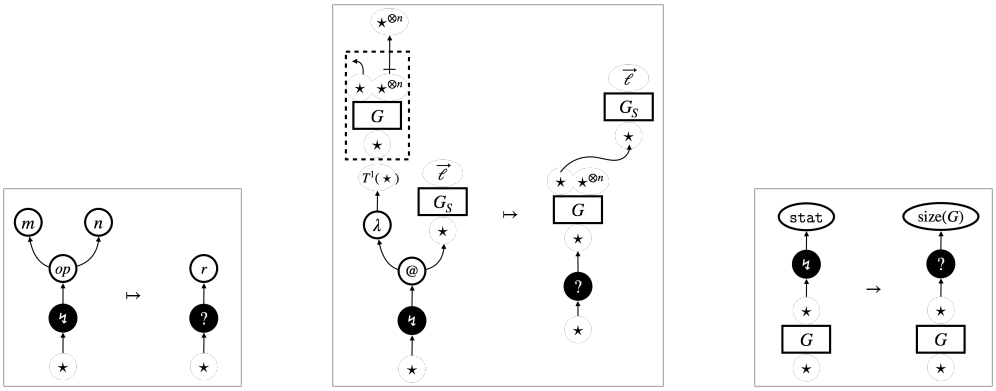


Figure 9: Interaction rules (a)–(e) and the substitution rule (f), where $t \in \{\checkmark, \lightning\}$ and G_S is a hypernet

Figure 9a: When the ?-focus encounters a variable (V), it changes to the $\lightning$-focus. What is connected above the variable will be substituted for the variable, in a subsequent substitute transition.

Figure 9b: When the ?-focus encounters an operation ϕ with at least one argument, it proceeds to the first argument.



(a) Arithmetic (b) Micro-beta (c) `stat`

Figure 10: The example behaviour $B_{\{+,-,\vec{@},\texttt{stat}\}}$ where $r = t\ op\ u$, $op \in \{+,-\}$ and G, G_S are hypernets

Figure 9c: After inspecting the $(m+1)$-th argument, the ✓-focus changes to the ?-focus and proceeds to the next argument.

Figure 9d: After inspecting all the arguments, the ✓-focus finishes redex search and changes to a focus depending on the operation ϕ_t: to the ✓-focus for a passive operation $\phi_\checkmark$, and to the $\lightning$-focus for an active operation $\phi_\lightning$.

Figure 9e: When the ?-focus encounters an operation that takes no arguments but only thunks, it immediately finishes redex search and changes to a focus depending on the operation ϕ_t, like in Figure 9d.

The first kind of rewrite transitions, namely substitution transitions, implements substitution by simply removing a variable edge (V). These transitions are specified by a *substitution rule* depicted in Figure 9f. What is connected above the variable edge (V) is computation bound to the variable. By removing the variable edge, the bound computation gets directly connected to the ?-focus, ready for redex search.

5.4. **Extrinsic transitions: behaviour of operations.** The second kind of rewrite transitions are for operations $\mathbb{O}$, in particular active operations $\mathbb{O}_\lightning$. These transitions are *extrinsic*; they are given as the second parameter $B_{\mathbb{O}}$, called *behaviour* of $\mathbb{O}$, of the UAM.

Figure 10 shows an example of the behaviour, namely that for the following active operations:

$$+ : \star \Rightarrow \star^{\otimes 2}, \quad - : \star \Rightarrow \star^{\otimes 2}, \quad \vec{@} : \star \Rightarrow \star^{\otimes 2}, \quad \texttt{stat} : \star \Rightarrow \epsilon.$$

For some of these operations, their behaviour is specified locally by *rewrite rules*. The rewrite transitions for the four active operations are enabled when the $\lightning$-focus encounters one of the operations. The $\lightning$-focus then changes to the ?-focus and resumes redex search.

Figure 10a: This rewrite rule specifies the behaviour of arithmetic $(+,-)$. The rule eliminates all three edges (m, n, op) and replace them with a new edge (r) such that $r = t\ op\ u$.

Figure 10b: This rewrite rule specifies the behaviour of application $(\vec{@})$, namely the (micro-) beta-reduction. It is *micro* in the sense that it delays substitution. It only eliminates

the constructors $(\lambda, @)$, opens the box whose content is G, and connects G_S which represents a function argument to the body G of the function.

Figure 10c: This is a rewrite transition, not a rewrite rule. It is for the operation `stat` that inspects memory usage. Namely, `stat` counts the number size(G) of edges in the hypernet G. The transition replaces the operation edge (`stat`) with the result (size(G)) of counting.

6. A formal definition of the UAM

The UAM, and hence the definitions below, are all globally parameterised by the operation set $\mathbb{O}$ and its behaviour $B_{\mathbb{O}}$.

6.1. **Auxiliary definitions.**

We use the terms *incoming* and *outgoing* to characterise the incidence relation between neighbouring edges. Conventionally incidence is defined relative to nodes, but we find it helpful to extend this notion to edges.

Definition 6.1 (Incoming and outgoing edges)**.** An *incoming* edge of an edge e has a target that is a source of the edge e. An *outgoing* edge of the edge e has a source that is a target of the edge e.

The notions of *path* and *reachability* are standard. Our technical development will heavily rely on these graph-theoretic notions. Note that these are the notions that are difficult to translate back into the language of terms.

Definition 6.2 (Paths and reachability)**.**

(1) A *path* in a hypergraph is given by a non-empty sequence of edges, where an edge e is followed by an edge e' if the edge e is an incoming edge of the edge e'.
(2) A vertex v' is *reachable* from a vertex v if $v = v'$ holds, or there exists a path from the vertex v to the vertex v'.

Note that, in general, the first edge (resp. the last edge) of a path may have no source (resp. target). A path is said to be *from* a vertex v, if v is a source of the first edge of the path. Similarly, a path is said to be *to* a vertex v', if v' is a target of the last edge of the path. A hypergraph G is itself said to be a path, if all edges of G comprise a path from an input (if any) and an output (if any) and every vertex is an endpoint of an edge.

During focussed hypernet rewriting, operations are the only edges that the ?-focus can "leave behind". The ?-focus is always at the end of an *operation path*.

Definition 6.3 (Operation paths)**.** A path whose edges are all labelled with operations is called *operation path*.

We shall introduce a few classes of hypernets below. The first is *box* hypernets that are simply single box edges.

Definition 6.4 (Box hypernets)**.** If a hypernet is a path of only one box edge, it is called *box* hypernet.

The second is *stable* hypernets, in which a focus can never trigger a rewrite (i.e. a focus never changes to the $\not{?}$-focus). Stable hypernets can be seen as a graph-based notion of values/normal form. For example, the hypernet that consists of an abstraction edge (λ) only is a stable hypernet.

Definition 6.5 (Stable hypernets)**.** A *stable* hypernet is a hypernet $(G : \star \Rightarrow \otimes_{i=1}^{m} \ell_i) \in \mathcal{H}(L_{\mathrm{lin}}, \mathbb{O}_{\checkmark})$, such that $\otimes_{i=1}^{m} \ell_i \in (\{T^n(\star) \mid n \in \mathbb{N}\})^m$ and each vertex is reachable from the unique input.

The last is *one-way* hypernets, which will play an important role in local reasoning. These specify sub-graphs to which a focus enters only from the bottom (i.e. the ?-focus through an input), never from the top (i.e. the ✓-focus through an output). Should the ✓-focus enter from the top, it must have traversed upwards the sub-graph and left an operation path behind. One-way hypernets are defined by ruling out such operation paths.

Definition 6.6 (One-way hypernets)**.** A hypernet H is *one-way* if, for any pair (v_i, v_o) of an input and an output of H such that v_i and v_o both have type $\star$, any path from v_i to v_o is not an operation path.

For example, the underlying hypernet H of the left hand side of the micro-beta rewrite rule (Figure 10b) is a one-way hypernet, if G_S is stable. Should the ✓-focus enters to H from the top, it must be backtracking the depth-first search, and hence the ?-focus must have been visited H from the bottom. In the presence of the micro-beta rewrite rule, such visit must result in a rewrite transition, and therefore, the backtracking of the ✓-focus cannot be possible.

6.2. **Focussed hypernets.** Focussed hypernets are those that contain a focus. We impose some extra conditions as below, to ensure that the focus is outside a box and not isolated.

Definition 6.7 (Focussed hypernets)**.**

(1) A focus in a hypergraph is said to be *exposed* if its source is an input and its target is an output, and *self-acyclic* if its source and its target are different vertices.
(2) *Focussed* hypernets (typically ranged over by $\dot{G}, \dot{H}, \dot{N}$) are those that contain only one focus and the focus is shallow, self-acyclic and not exposed.

Focus-free hypernets are given by $\mathcal{H}_{\omega}(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O})\backslash\{?, \checkmark, \lightning\})$, i.e. hypernets without a focus.

Notation 6.8 (Removing, replacing and attaching a focus)**.**

(1) A focussed hypernet $\dot{G}$ can be turned into an *underlying* focus-free hypernet $|\dot{G}|$ with the same type, by removing its unique focus and identifying the source and the target of the focus.
(2) When a focussed hypernet $\dot{G}$ has a $\mathtt{t}$-focus, then changing the focus label $\mathtt{t}$ to another one $\mathtt{t}'$ yields a focussed hypernet denoted by $\langle \dot{G} \rangle_{\mathtt{t}'/\mathtt{t}}$.
(3) Given a focus-free hypernet G, a focussed hypernet $\mathtt{t};_i G$ with the same type can be yielded by connecting a $\mathtt{t}$-focus to the i-th input of G if the input has type $\star$. Similarly, a focussed hypernet $G;_i \mathtt{t}$ with the same type can be yielded by connecting a $\mathtt{t}$-focus to the i-th output of G if the output has type $\star$. If it is not ambiguous, we omit the index i in the notation $;_i$.

The source (resp. target) of a focus is called "focus source" (resp. "focus target") in short.

6.3. **Contexts.** We next formalise a notion of *context*, which is hypernets with *holes*. We use a set $\mathbb{M}$ of hole labels, and contexts are allowed to contain an arbitrary number of holes. Hole labels are typed, and typically ranged over by $\chi : \vec{\ell} \Rightarrow \vec{\ell'}$.

Definition 6.9 ((Simple) contexts)**.**

(1) *Holed* hypernets (typically ranged over by $\mathcal{C}$) are given by $\mathcal{H}_\omega(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O}) \cup \mathbb{M})$, where the edge label set $M_{\mathrm{lin}}(\mathbb{O})$ is extended by the set $\mathbb{M}$.
(2) A holed hypernet $\mathcal{C}$ is said to be *context* if each hole label appears at most once (at any depth) in $\mathcal{C}$.
(3) A *simple* context is a context that contains a single hole, which is shallow.

By what we call *plugging*, we can replace a hole of a context with a hypernet, and obtain a new context. We here provide a description of plugging and fix a notation. A formal definition of plugging can be found in Appendix B.

Notation 6.10 (Plugging of contexts)**.**

(1) When $\vec{\chi}$ gives a list of all and only hole labels that appear in a context $\mathcal{C}$, the context can also be written as $\mathcal{C}[\vec{\chi}]$. A hypernet in $\mathcal{H}_\omega(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O}))$ can be seen as a context without a hole and written as $\mathcal{C}[\,]$.
(2) Let $\mathcal{C}[\vec{\chi^1}, \chi, \vec{\chi^2}]$ and $\mathcal{C}'[\vec{\chi^3}]$ be contexts, such that the hole χ and the latter context $\mathcal{C}'$ have the same type and $\vec{\chi^1} \cap \vec{\chi^2} \cap \vec{\chi^3} = \emptyset$. A new context $\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^2}] \in \mathcal{H}_\omega(L_{\mathrm{lin}}, M_{\mathrm{lin}} \cup \vec{\chi^1} \cup \vec{\chi^3} \cup \vec{\chi^2}))$ can be obtained by *plugging* $\mathcal{C}'$ into $\mathcal{C}$: namely, by replacing the (possibly deep) hole edge of $\mathcal{C}$ that has label χ with the context $\mathcal{C}'$, and by identifying each input (resp. output) of $\mathcal{C}'$ with its corresponding source (resp. target) of the hole edge.

Each edge of the new context $\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^3}]$ is inherited from either $\mathcal{C}$ or $\mathcal{C}'$, keeping the type; this implies that the new context is indeed a context with hole labels $\vec{\chi^1}, \vec{\chi^3}, \vec{\chi^2}$. Inputs and outputs of the new context coincide with those of the original context $\mathcal{C}$, and hence these two contexts have the same type.

The plugging is associative in two senses: plugging two contexts into two holes of a context yields the same result regardless of the order, i.e. $\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^2}, \mathcal{C}'', \vec{\chi^3}]$ is well-defined; and nested plugging yields the same result regardless of the order, i.e. $\mathcal{C}[\vec{\chi^1}, \mathcal{C}'[\vec{\chi^3}, \mathcal{C}'', \vec{\chi^4}], \vec{\chi^2}] = (\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^2}])[\vec{\chi^1}, \vec{\chi^3}, \mathcal{C}'', \vec{\chi^4}, \vec{\chi^2}]$.

The notions of focussed and focus-free hypernets can be naturally extended to contexts. We use the terms *entering* and *exiting* to refer to a focus that is adjacent to a hole. A focus may be both entering and exiting.

Definition 6.11 (Entering/exiting focuses)**.** In a focussed context $\dot{\mathcal{C}}[\vec{\chi}]$, the focus is said to be *entering* if it is an incoming edge of a hole, and *exiting* if it is an outgoing edge of a hole.

6.4. **States and transitions.** We now define the UAM as a state transition system. States are hypernets that represent closed terms and hence have type $\star \Rightarrow \epsilon$.

Definition 6.12 (States)**.**

(1) A *state* is given by a focussed hypernet $\dot{G} \in \mathcal{H}_\omega(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O}))$ of type $\star \Rightarrow \epsilon$.
(2) A state $\dot{G}$ is called *initial* if $\dot{G} = ?; |\dot{G}|$, and *final* if $\dot{G} = \checkmark; |\dot{G}|$.

The following will be apparent once transitions are defined: initial states are indeed initial in the sense that no search transition results in an initial state; and final states are indeed final in the sense that no transition is possible from a final state.

Intrinsic transitions, which consists of search transitions and substitution transitions, are specified by the interaction rules and the substitution rule in Figure 9. Each intrinsic transition applies a rule outside a box and at one place.

Definition 6.13 (Intrinsic transitions)**.**

(1) For each interaction rule $\dot{G} \overset{\bullet}{\mapsto} \dot{G}'$, if there exists a focus-free simple context $\mathcal{C}[\chi] : \star \Rightarrow \epsilon$ such that $\mathcal{C}[\dot{G}]$ and $\mathcal{C}[\dot{G}']$ are states, $\mathcal{C}[\dot{G}] \to \mathcal{C}[\dot{G}']$ is a *search transition*.

(2) For each substitution rule $\dot{G} \overset{\vee}{\mapsto} \dot{G}'$, if there exists a focus-free simple context $\mathcal{C}[\chi] : \star \Rightarrow \epsilon$ such that $\mathcal{C}[\dot{G}]$ and $\mathcal{C}[\dot{G}']$ are states, $\mathcal{C}[\dot{G}] \to \mathcal{C}[\dot{G}']$ is a *substitution transition*.

When a sequence $\dot{G} \to^* \dot{G}'$ of transitions consists of search transitions only, it is annotated by the symbol $\bullet$ as $\dot{G} \overset{\bullet}{\to}^* \dot{G}'$.

Extrinsic transitions $B_{\mathbb{O}}$ must have a specific form; namely they must be a *compute transition*.

Definition 6.14 (Compute transitions)**.** A transition $\dot{G} \to \dot{G}'$ is a *compute transition* if: (i) the first state $\dot{G}$ has the $\lightning$-focus that is an incoming edge of an active operation edge; and (ii) the second state $\dot{G}'$ has the ?-focus.

We can observe that substitution transitions or compute transitions are possible if and only if a state has the $\lightning$-focus, and they always change the focus to the ?-focus. We refer to substitution transitions and compute transitions altogether as *rewrite* transitions (cf. Table 1).

Compute transitions may be specified locally, by *rewrite rules*, in the same manner as the intrinsic transitions. Figure 10a & 10b shows examples of rewrite rules. We leave it entirely open what the actual rewrite associated to some operation is, by having the behaviour $B_{\mathbb{O}}$ as parameter of the UAM as well as the operation set $\mathbb{O}$. This is part of the semantic flexibility of our framework. We do not specify a meta-language for encoding effects as particular transitions. Any *algorithmic* state transformation (e.g. the compute transition for `stat` in Figure 10c) is acceptable.

We can now define the UAM as follows.

Definition 6.15 (the UAM)**.** Given two parameters $\mathbb{O}$ and $B_{\mathbb{O}}$, the *universal abstract machine (UAM)* $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is given by data $(S_{\mathbb{O}}, T \uplus B_{\mathbb{O}})$ such that:

- $S_{\mathbb{O}} \subseteq \mathcal{H}_\omega(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O}))$ is a set of states,
- $T \subseteq S_{\mathbb{O}} \times S_{\mathbb{O}}$ is a set of intrinsic transitions, and
- $B_{\mathbb{O}} \subseteq S_{\mathbb{O}} \times S_{\mathbb{O}}$ is a set of compute transitions.

We refer to elements of $B_{\mathbb{O}}$ as *extrinsic* transitions, as well as *compute* transitions; we use these two terms interchangeably.

An *execution* of the UAM starts with a focus-free hypernet that represents a closed term, e.g. a result of the translation $(-)^\dagger$ (cf. Figure 6). It is successful if it terminates with a final state, and not if it gets stuck with a non-final state.

Definition 6.16 (Execution and stuck states)**.**

(1) An *execution* on a focus-free hypernet $G : \star \Rightarrow \epsilon$ is a sequence of transitions starting from the initial state $?; G$.

(2) A state is said to be *stuck* if it is not final and cannot be followed by any transition.

Recall that we have a notion of *stable hypernet* (Definition 6.5) that is a graphical counterpart of values. An execution on any stable hypernet terminates successfully at a final state, with only search transitions (cf. Lemma D.6(1) which is proved for the "non-linear" UAM).

7. Contextual equivalence on hypernets

In this section, we set the target of our proof methodology. First, we define contextual equivalence in a general manner. Next, we clarify what kind of operations $\mathbb{O}$ our proof methodology applies to.

7.1. Generalised contextual equivalence.

We propose notions of contextual refinement and equivalence that check for successful termination of execution[4]. Our notion of contextual refinement (and hence contextual equivalence) generalise the standard notions in two ways.

- The notion of contextual refinement can be flexible in terms of a class of contexts in which it holds. Namely, contextual refinement is parameterised by a set $\mathbb{C} \subseteq \mathcal{H}_\omega(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O}) \cup \mathbb{M})$ of focus-free contexts. The standard contextual refinement can be recovered by setting $\mathbb{C}$ to be the set of all focus-free contexts.
- The notion of contextual refinement can count and compare the number of transitions. Namely, contextual refinement is parameterised by a preorder Q on natural numbers. The standard contextual refinement can be obtained by setting Q to be the total relation $\mathbb{N} \times \mathbb{N}$. Other typical examples of the preorder Q are the greater-than-or-equal relation $\geq_{\mathbb{N}}$ and the equality $=_{\mathbb{N}}$. With these preorders, one can prove that two terms are contextually equivalent, and moreover, one takes a less number of transitions to terminate than the other (with $\geq_{\mathbb{N}}$) or the two terms take exactly the same number of transitions to terminate (with $=_{\mathbb{N}}$).

We require the parameter $\mathbb{C}$ to be closed under plugging, i.e. for any contexts $\mathcal{C}[\vec{\chi^1}, \chi, \vec{\chi^2}]$, $\mathcal{C}' \in \mathbb{C}$ such that $\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^2}]$ is defined, $\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^2}] \in \mathbb{C}$.

Definition 7.1 (State refinement and equivalence)**.** Let Q be a preorder on $\mathbb{N}$, and $\dot{G}_1$ and $\dot{G}_2$ be two states.

- $\dot{G}_1$ is said to *refine* $\dot{G}_2$ up to Q, written as $B_{\mathbb{O}} \models (\dot{G}_1 \mathrel{\dot{\preceq}_Q} \dot{G}_2)$, if for any number $k_1 \in \mathbb{N}$ and any final state $\dot{N}_1$ such that $\dot{G}_1 \to^{k_1} \dot{N}_1$, there exist a number $k_2 \in \mathbb{N}$ and a final state $\dot{N}_2$ such that $k_1 \; Q \; k_2$ and $\dot{G}_2 \to^{k_2} \dot{N}_2$.
- $\dot{G}_1$ and $\dot{G}_2$ are said to be *equivalent* up to Q, written as $B_{\mathbb{O}} \models (\dot{G}_1 \mathrel{\dot{\simeq}_Q} \dot{G}_2)$, if $B_{\mathbb{O}} \models (\dot{G}_1 \mathrel{\dot{\preceq}_Q} \dot{G}_2)$ and $B_{\mathbb{O}} \models (\dot{G}_2 \mathrel{\dot{\preceq}_Q} \dot{G}_1)$.

Definition 7.2 (Contextual refinement and equivalence)**.** Let $\mathbb{C}$ be a set of contexts that is closed under plugging, Q be a preorder on $\mathbb{N}$, and H_1 and H_2 be focus-free hypernets of the same type.

[4]We opt for the very basic notion of contextual refinement that concerns termination only. Richer observation (e.g. evaluation results, output, probability, nondeterminism) would require different definitions of contextual refinement, and these are out of the scope of this paper.

- H_1 is said to *contextually refine* H_2 in $\mathbb{C}$ up to Q, written as $B_{\mathbb{O}} \models (H_1 \preceq^{\mathbb{C}}_{Q} H_2)$, if any focus-free context $\mathcal{C}[\chi] \in \mathbb{C}$, such that $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_2]$ are states, yields refinement $B_{\mathbb{O}} \models (?;\mathcal{C}[H_1] \mathrel{\dot\preceq_Q} ?;\mathcal{C}[H_2])$.
- H_1 and H_2 are said to be *contextually equivalent* in $\mathbb{C}$ up to Q, written as $B_{\mathbb{O}} \models (H_1 \simeq^{\mathbb{C}}_{Q} H_2)$, if $B_{\mathbb{O}} \models (H_1 \preceq^{\mathbb{C}}_{Q} H_2)$ and $B_{\mathbb{O}} \models (H_2 \preceq^{\mathbb{C}}_{Q} H_1)$.

In the sequel, we simply write $\dot G_1 \mathrel{\dot\preceq_Q} \dot G_2$ etc., making the parameter $B_{\mathbb{O}}$ implicit.

Because Q is a preorder, $\dot\preceq_Q$ and $\preceq^{\mathbb{C}}_{Q}$ are indeed preorders, and accordingly, equivalences $\dot\simeq_Q$ and $\simeq^{\mathbb{C}}_{Q}$ are indeed equivalences (Lemma E.2).

When the relation Q is the universal relation $\mathbb{N} \times \mathbb{N}$, the notions concern successful termination, and the number of transitions is irrelevant. If all compute transitions are deterministic, contextual equivalences $\simeq^{\mathbb{C}}_{\geq_{\mathbb{N}}}$ and $\simeq^{\mathbb{C}}_{=_{\mathbb{N}}}$ coincide for any $\mathbb{C}$ (as a consequence of Lemma E.3).

Because $\mathbb{C}$ is closed under plugging, the contextual notions $\preceq^{\mathbb{C}}_{Q}$ and $\simeq^{\mathbb{C}}_{Q}$ indeed become congruences. Namely, for any $H_1 \mathrel{\square^{\mathbb{C}}} H_2$ and $\mathcal{C} \in \mathbb{C}$ such that $\mathcal{C}[H_1]$ and $\mathcal{C}[H_2]$ are defined, $\mathcal{C}[H_1] \mathrel{\square^{\mathbb{C}}} \mathcal{C}[H_2]$, where $\square \in \{\preceq_Q, \simeq_Q\}$.

As the parameter $\mathbb{C}$, we will use the set $\mathbb{C}_{\mathbb{O}}$ of all focus-free contexts, for the time being. We will use another set in Section 11. The standard notions of contextual refinement and equivalence can be recovered as $\preceq^{\mathbb{C}_{\mathbb{O}}}_{\mathbb{N}\times\mathbb{N}}$ and $\simeq^{\mathbb{C}_{\mathbb{O}}}_{\mathbb{N}\times\mathbb{N}}$.

7.2. **Determinism and refocusing.** We will focus on operations $\mathbb{O}$ whose behaviour $B_{\mathbb{O}}$ makes the UAM both *deterministic* and *refocusing* in the following sense.

Definition 7.3 (Determinism and refocusing)**.**

(1) A UAM $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is *deterministic* if the following holds: if two transitions $\dot G \to \dot G'$ and $\dot G \to \dot G''$ are possible, it holds that $\dot G' = \dot G''$ up to graph isomorphism.
(2) A state $\dot G$ is *rooted* if $?;|\dot G| \xrightarrow{\bullet}{}^{*} \dot G$.
(3) A UAM $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is *refocusing* if every transition preserves the rooted property.

In a refocusing UAM, the rooted property becomes an invariant, because any initial state is trivially rooted. The invariant ensures the following: starting the search process (i.e. a search transition) from a state $\dot N'$ with the ?-focus can be seen as *resuming* the search process $?;|\dot N'| \xrightarrow{\bullet}{}^{*} \dot N'$, from an initial state, on the underlying hypernet $|\dot N'|$. Resuming redex search after a rewrite, rather than starting from scratch, is an important aspect of abstract machines. In the case of the lambda-calculus, enabling the resumption is identified as one of the key steps (called "refocusing") to synthesise abstract machines from reduction semantics by Danvy et al. [DMMZ12]. In our setting, it is preservation of the rooted property that justifies the resumption.

For many rewrite transitions that are specified by a local rewrite rule, it suffices to check the shape of the rewrite rule, in order to conclude that the corresponding rewrite transition preserves the rooted property. Intuitively, a rewrite rule $\dot H \mapsto \dot H'$ (or the substitution rule) should satisfy the following.

(1) No $\checkmark$-focus can encounter $|\dot H|$ prior to application of the rewrite rule $\dot H \mapsto \dot H'$.
(2) The rewrite rule changes only edges above the $\lightning$-focus, and turns the focus into the ?-focus without moving it around.
(3) If the ?-focus encounters $|\dot H|$, subsequent search transitions yield the $\lightning$-focus.

The above idea is formalised as a notion of *stationary* rewrite transition. If a rewrite transition is *stationary* in the following sense, it preserves the rooted property.

Definition 7.4 (Stationary rewrite transitions)**.** A rewrite transition $\dot{G} \to \dot{G}'$ is *stationary* if there exist a focus-free simple context $\mathcal{C}$, focus-free hypernets H and H', and a number $i \in \mathbb{N}$, such that the following holds.

(1) H is one-way,
(2) $\dot{G} = \mathcal{C}[\lightning ;_i H]$ and $\dot{G}' = \mathcal{C}[?;_i H']$, and
(3) for any $j \in \mathbb{N}\backslash\{i\}$, such that $\mathcal{C}[?;_j H]$ is a state, there exists a state $\dot{N}$ with the $\lightning$-focus, such that $\mathcal{C}[?;_j H] \xrightarrow{\bullet} \dot{N}$.

Lemma 7.5 (Lemma C.8)**.** *If a rewrite transition $\dot{G} \to \dot{G}'$ is stationary, it preserves the rooted property, i.e. $\dot{G}$ being rooted implies $\dot{G}'$ is also rooted.*

Finally, determinism and refocusing of a UAM boil down to those of extrinsic transitions $B_{\mathbb{O}}$ under a mild condition.

Lemma 7.6 (Determinism and refocusing)**.**

- *A universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is deterministic if extrinsic transitions $B_{\mathbb{O}}$ are deterministic.*
- *Suppose that G_S in the substitution rule (Figure 9f) is a stable hypernet. A universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is refocusing if extrinsic transitions $B_{\mathbb{O}}$ preserve the rooted property.*

Proof. Intrinsic transitions and extrinsic transitions are mutually exclusive, and intrinsic transitions are all deterministic for the following reasons.

- Search transitions are deterministic, because at most one interaction rule can be applied at any state.
- Although two different substitution rules may be possible at a state, substitution transitions are still deterministic. Namely, if two different substitution rules $\dot{G} \overset{\vee}{\mapsto} \dot{G}'$ and $\dot{H} \overset{\vee}{\mapsto} \dot{H}'$ can be applied to the same state, i.e. there exist focus-free simple contexts $\mathcal{C}_G$ and $\mathcal{C}_H$ such that $\mathcal{C}_G[\dot{G}] = \mathcal{C}_H[\dot{H}]$, then these two rules yield the same transition, by satisfying $\mathcal{C}_G[\dot{G}'] = \mathcal{C}_H[\dot{H}']$.

Therefore, if extrinsic transitions are deterministic, all transitions become deterministic.

Search transitions trivially preserve the rooted property. Substitution transitions also preserve the rooted property, because they are stationary, under the assumption that G_S in Figure 9f is stable. Therefore, all transitions but extrinsic transitions already preserve the rooted property. □

8. A sufficiency-of-robustness theorem

In this section, we will state a *sufficiency-of-robustness theorem* (Theorem 8.11), by identifying sufficient conditions (namely *safety* and *robustness*) for contextual refinement $N \preceq^{\mathbb{C}}_{Q} H$. Throughout this section, we will use the micro-beta law, which looks like Figure 11, as a leading example. It is derived from the micro-beta rewrite rule (Figure 10b) by removing focuses, and it is the core of a graphical counterpart of the beta-law $(\lambda x.t)\ v \preceq t[v/x]$. We will introduce relevant notions and concepts (such as safety and robustness), by illustrating what it takes for the micro-beta law to hold.

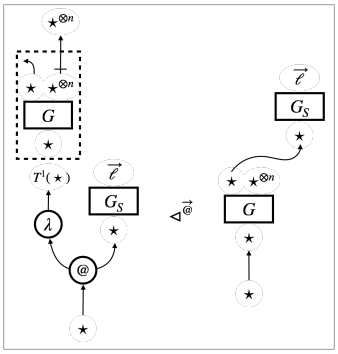


Figure 11: The micro-beta pre-template, where G is a hypernet and G_S is a stable hypernet

8.1. **Pre-templates and specimens.** We begin with an observation that we are often interested in a *family* of contextual refinements. Syntactically, indeed, the beta-law $(\lambda x.t)\ v \preceq t[v/x]$ represents a family of (concrete) contextual refinements such as $(\lambda x.x)\ (\lambda y.y) \preceq \lambda y.y$. Graphically, it is the same; the micro-beta law represents a family of (concrete) laws. In the micro-beta law (Figure 11), G can be arbitrary and G_S can be any stable hypernet. The sufficiency-of-robustness theorem will therefore take a family $\lhd$ of pairs of focus-free hypernets (i.e. a relation $\lhd$ on focus-free hypernets), and identifies sufficient conditions for the family to imply a family of (concrete) contextual refinements.

Moreover, the relation $\lhd$ must be well-typed, i.e. each pair $(N, H) \in \lhd$ must share the same type. This in particular means that, if N represents a term (or a thunk), H must represents a term (or resp. a thunk) with the same number of free variables. We therefore formalise $\lhd$ as a type-indexed family of relations on focus-free hypernets, and call it *pre-template*. A pre-template is our candidate of (a family of) contextual refinement.

Definition 8.1 (Pre-templates)**.** A *pre-template* is given by a union $\lhd := \cup_{I \in \mathcal{I}} \lhd_I$ of a type-indexed family $\{\lhd_I\}_{I \in \mathcal{I}}$, where $\mathcal{I}$ is a set of types. Each $\lhd_I$ is a binary relation on focus-free hypernets such that, for any $G_1 \lhd_I G_2$ where $I \in \mathcal{I}$, G_1 and G_2 are focus-free hypernets with type $G_1 : I$ and $G_2 : I$.

Example 8.2 (Micro-beta pre-template $\lhd^{\vec{@}}$)**.** As a leading example, we consider the *micro-beta* pre-template $\lhd^{\vec{@}}$, depicted in Figure 11, derived from the micro-beta rewrite rule (Figure 10b). The pre-template additionally requires G_S to be stable, compared to the micro-beta rewrite rule; this amounts to require v to be a value in the beta-law $(\lambda x.t)\ v \preceq t[v/x]$. For each $N \lhd^{\vec{@}} H$, the hypernets N and H have the same type $\star \Rightarrow \star^{\otimes n} \otimes \vec{\ell}$, where $n \in \mathbb{N}$ and the sequence $\vec{\ell}$ of types can be arbitrary as long as G_S is stable. ■

To directly prove that a pre-template $N \lhd H$ implies contextual refinement $N \preceq^{\mathbb{C}}_{Q} H$, one would need to compare states $\dot{C}[N], \dot{C}[H]$ for any focussed context $\dot{C}$. We use data dubbed *specimen* to provide such a pair. Sometimes we prefer relaxed comparison, between two states $\dot{P}_1, \dot{P}_2$ such that $\dot{P}_1\ R\ \dot{C}[N]$ and $\dot{C}[H]\ R'\ \dot{P}_2$ for some binary relations R, R' on states. Such comparison can be specified by what we call *quasi-specimen up to* (R, R'). The notion of (quasi-)specimen is relative to the set $\mathbb{C}$ of focus-free contexts, which is one of the parameters of contextual refinement.

Definition 8.3 ((Quasi-)specimens)**.** Let $\lhd$ be a pre-template, and R and R' be binary relations on states.

(1) A triple $(\dot{\mathcal{C}}[\vec{\chi}]; \vec{H}^1; \vec{H}^2)$ is a $\mathbb{C}$-*specimen* of $\lhd$ if the following hold:
   (A) $|\dot{\mathcal{C}}[\vec{\chi}]| \in \mathbb{C}$, and the three sequences $\vec{\chi}, \vec{H}^1, \vec{H}^2$ have the same length n.
   (B) $H_i^1 \lhd H_i^2$ for each $i \in \{1, \ldots, n\}$.
   (C) $\dot{\mathcal{C}}[\vec{H}^p]$ is a state for each $p \in \{1, 2\}$.

(2) A pair $(\dot{N}_1, \dot{N}_2)$ of states is a *quasi-*$\mathbb{C}$*-specimen* of $\lhd$ *up to* (R, R'), if there exists a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$ of $\lhd$ such that the following hold:
   (A) The focuses of $\dot{\mathcal{C}}$, $\dot{N}_1$ and $\dot{N}_2$ all have the same label.
   (B) If $\dot{N}_1$ and $\dot{N}_2$ are rooted, then $\dot{\mathcal{C}}[\vec{H}^1]$ and $\dot{\mathcal{C}}[\vec{H}^2]$ are also rooted, $\dot{N}_1 \; R \; \dot{\mathcal{C}}[\vec{H}^1]$, and $\dot{\mathcal{C}}[\vec{H}^2] \; R' \; \dot{N}_2$.

(3) A $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\vec{\chi}]; \vec{H}^1; \vec{H}^2)$ is said to be *single* if the sequence $\vec{\chi}$ only has one element, i.e. the context $\dot{\mathcal{C}}$ has exactly one hole edge (at any depth).

We can refer to *the* focus label of a $\mathbb{C}$-specimen and a quasi-$\mathbb{C}$-specimen. Any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$ gives a quasi-$\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\vec{H}^1], \dot{\mathcal{C}}[\vec{H}^2])$ up to $(=, =)$.

As described in Section 2.3, the key part of proving contextual refinement $N \preceq_Q^{\mathbb{C}} H$ for each $N \lhd H$ is to show (2.2). With the notion of specimen in hand, (2.2) can be rephrased as follows:

- for any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\vec{\chi}]; \vec{N}; \vec{H})$ of $\lhd$ and a transition $\dot{\mathcal{C}}[\vec{N}] \to \dot{P}$,
- there exist a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'[\vec{\chi}']; \vec{N}'; \vec{H}')$ and $k, l \in \mathbb{N}$ such that $(1+k) \; Q \; l$ and the following holds.

$$\begin{array}{ccc} \dot{\mathcal{C}}[\vec{N}] & \longrightarrow \dot{P} \xrightarrow{k} & \dot{\mathcal{C}}'[\vec{N}'] \\ \rotatebox{90}{$\lhd$} & & \rotatebox{90}{$\lhd$} \\ \dot{\mathcal{C}}[\vec{H}] & \xrightarrow{\quad\quad\quad\quad\quad} _{l} & \dot{\mathcal{C}}'[\vec{H}'] \end{array} \tag{8.1}$$

To establish (8.1), it suffices to perform case analysis on the transition $\dot{\mathcal{C}}[\vec{N}] \to \dot{P}$. As explained in Section 2.3, there are three possible cases. The first case is a trivial case, where the ?-focus or ✓-focus moves just inside the context $|\dot{\mathcal{C}}|$. In this case, we can take the updated context $\dot{\mathcal{C}}'$ such that $|\dot{\mathcal{C}}| = |\dot{\mathcal{C}}'|$. For the other two cases, we identify sufficient conditions for (8.1), namely safety and robustness.

8.2. **Safety.** The second case of case analysis for (8.1) is when the ?-focus or the ✓-focus encounters one of $\vec{N}$ (and $\vec{H}$); see Figure 4b. Let us look at how the micro-beta pre-template $\lhd^{\vec{@}}$ (cf. Figure 10b; mind that the focuses are dropped in the pre-template) satisfies (8.1) in this case. Let $N_i \lhd^{\vec{@}} H_i$ be the ones the focus is encountering. There are three sub-cases.

Case (ii-1) Searching: The ?-focus enters N_i, H_i, i.e. we have $?; N_i$ and $?; H_i$ inside states $\dot{\mathcal{C}}[\vec{N}], \dot{\mathcal{C}}[\vec{H}]$. On $?; N_i$, a few search transitions will be followed by a compute transition that applies the micro-beta rewrite rule, because what is connected to the right target of the application edge (@) (i.e. G_S in Figure 11) is required to be stable. The result is exactly $?; H_i$.

This means that we can take a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'[\vec{\chi}']; \vec{N}'; \vec{H}')$ where: $\dot{\mathcal{C}}'$ is obtained by replacing the i-th hole of $\dot{\mathcal{C}}$ with H_i; and $\vec{N}', \vec{H}'$ are obtained by removing N_i, H_i from

$\vec{N}, \vec{H}$. One observation is that the focus in $\dot{\mathcal{C}}'$ is not entering (i.e. pointing at a hole), because the focus is the ?-focus pointing at H_i.

Case (ii-2) Backtracking on the box: The ✓-focus is on top of the box (i.e. G in Figure 11). This case is in fact impossible in a refocusing UAM. To make states $\dot{\mathcal{C}}[\vec{N}], \dot{\mathcal{C}}[\vec{H}]$ rooted, the ✓-focus must be at the end of an operation path. However, the ✓-focus in the state $\dot{\mathcal{C}}[\vec{N}]$ is adjacent to a box edge, and cannot be at the end of an operation path.

Case (ii-3) Backtracking on the stable argument: The ✓-focus is on top of what is connected to the right target of the application edge (@) (i.e. G_S in Figure 11). This case is also impossible, because of types. The stable hypernet G_S has type $\star \Rightarrow \otimes_{i=1}^{m} T^{n_i}(\star)$ (cf. Definition 6.5), and the ✓-focus has type $\star \Rightarrow \star$. The ✓-focus cannot be on top of G_S.

From Case (ii-1), we extract a sufficient condition for (8.1), dubbed *input-safety*, as follows. The micro-beta pre-template $\lhd^{\vec{@}}$ falls into (II) below.

Definition 8.4 (Input-safety)**.** A pre-template $\lhd$ is $(\mathbb{C}, Q, Q')$-*input-safe* if, for any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$ of $\lhd$ such that $\dot{\mathcal{C}}$ has the entering ?-focus, one of the following holds.
(I) There exist two stuck states $\dot{N}_1$ and $\dot{N}_2$ such that $\dot{\mathcal{C}}[\vec{H}^p] \to^* \dot{N}_p$ for each $p \in \{1, 2\}$.
(II) There exist a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'}^1; \vec{H'}^2)$ of $\lhd$ and two numbers $k_1, k_2 \in \mathbb{N}$, such that the focus of $\dot{\mathcal{C}}'$ is the ✓-focus or the non-entering ?-focus, $(1 + k_1)\ Q\ k_2$, $\dot{\mathcal{C}}[\vec{H}^1] \to^{1+k_1} \dot{\mathcal{C}}'[\vec{H'}^1]$, and $\dot{\mathcal{C}}[\vec{H}^2] \to^{k_2} \dot{\mathcal{C}}'[\vec{H'}^2]$.
(III) There exist a quasi-$\mathbb{C}$-specimen $(\dot{N}_1, \dot{N}_2)$ of $\lhd$ up to $(\dot{\simeq}_{Q'}, \dot{\simeq}_{Q'})$, whose focus is not the $\lightning$-focus, and two numbers $k_1, k_2 \in \mathbb{N}$, such that $(1 + k_1)\ Q\ (1 + k_2)$, $\dot{\mathcal{C}}[\vec{H}^1] \to^{1+k_1} \dot{N}_1$, and $\dot{\mathcal{C}}[\vec{H}^2] \to^{1+k_2} \dot{N}_2$.

Case (ii-2) and (ii-3) tells us that types and the rooted property prevents the ✓-focus from visiting N_i, H_i such that $N_i \lhd^{\vec{@}} H_i$. This situation can be captured by one-way hypernets (Definition 6.6), resulting in another sufficient condition dubbed *output-closure*.

Definition 8.5 (Output-closure)**.** A pre-template $\lhd$ is *output-closed* if, for any hypernets $H_1 \lhd H_2$, either H_1 or H_2 is one-way.

Input-safety and output-closure are the precise safety conditions. When a pre-template is safe, we simply call it *template*.

Definition 8.6 (Templates)**.** A pre-template $\lhd$ is a $(\mathbb{C}, Q, Q')$-*template*, if it is $(\mathbb{C}, Q, Q')$-input-safe and also output-closed.

The micro-beta pre-template $\lhd^{\vec{@}}$ is indeed safe, and hence a template. The third parameter of input-safety is irrelevant here, and we can simply set it as the equality $=$. Recall that $\mathbb{C}_{\mathbb{O}}$ is the set of all focus-free contexts using operations $\mathbb{O}$.

Lemma 8.7 (Safety of $\lhd^{\vec{@}}$)**.** *Let* $\mathbb{O}^{\mathrm{lin}}_{\checkmark}$ *be the set* $\mathbb{N} \cup \{\lambda\}$, $\mathbb{O}^{\mathrm{lin}}_{\lightning}$ *be the set* $\{+, -, \vec{@}, \mathtt{stat}\}$, *and* $\mathbb{O}^{\mathrm{lin}} = \mathbb{O}^{\mathrm{lin}}_{\checkmark} \uplus \mathbb{O}^{\mathrm{lin}}_{\lightning}$. *The micro-beta pre-template* $\lhd^{\vec{@}}$ *is a* $(\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}, \geq, =)$-*template.* □

8.3. **Robustness.** The last case of case analysis for (8.1) is when the $\lightning$-focus triggers a rewrite transition in $\dot{\mathcal{C}}$ (and hence in $\dot{\mathcal{C}}'$). Does the micro-beta template satisfy (8.1) in this case? It depends on the rewrite transition $\dot{\mathcal{C}}[\vec{N}] \to \dot{P}$. Let us consider an operation set $\mathbb{O}^{\text{lin}} = \mathbb{O}^{\text{lin}}_{\checkmark} \uplus \mathbb{O}^{\text{lin}}_{\lightning}$ where $\mathbb{O}^{\text{lin}}_{\checkmark} = \mathbb{N} \cup \{\lambda\}$ and $\mathbb{O}^{\text{lin}}_{\lightning} = \{+, -, \vec{@}, \texttt{stat}\}$. We can perform case analysis in terms of which rewrite is triggered by the $\lightning$-focus.

Case (iii-1) Substitution: The $\lightning$-focus triggers application of the substitution rule (Figure 9f). The rule simply removes a variable edge (V), and the same rule can be applied to both states $\dot{\mathcal{C}}[\vec{N}], \dot{\mathcal{C}}[\vec{H}]$. Sub-graphs related by $\lhd^{\vec{@}}$ may be involved, as part of G_S in Figure 9f, but they are kept unchanged. Therefore we can take a $\mathbb{C}_{\mathbb{O}^{\text{lin}}}$-specimen $(\dot{\mathcal{C}}'[\vec{\chi}']; \vec{N}'; \vec{H}')$ where $\dot{\mathcal{C}}'$ has the same number of holes as $\dot{\mathcal{C}}$.

Case (iii-2) Arithmetic: The $\lightning$-focus triggers application of the arithmetic rewrite rule (Figure 10a). The rewrite rule only involves three edges, and it can never involves sub-graphs related by $\lhd^{\vec{@}}$. We can also take a $\mathbb{C}_{\mathbb{O}^{\text{lin}}}$-specimen $(\dot{\mathcal{C}}'[\vec{\chi}']; \vec{N}'; \vec{H}')$ where $\dot{\mathcal{C}}'$ has the same number of holes as $\dot{\mathcal{C}}$.

Case (iii-3) Micro-beta: The $\lightning$-focus triggers application of the micro-beta rewrite rule (Figure 10b). Sub-graphs related by $\lhd^{\vec{@}}$ may be involved in the rewrite rule, as part of the box content G in Figure 10b, but they are kept unchanged. We can also take a $\mathbb{C}_{\mathbb{O}^{\text{lin}}}$-specimen $(\dot{\mathcal{C}}'[\vec{\chi}']; \vec{N}'; \vec{H}')$ where $\dot{\mathcal{C}}'$ has the same number of holes as $\dot{\mathcal{C}}$.

Case (iii-4) `stat`: The $\lightning$-focus triggers application of the `stat` rewrite rule (Figure 10c). The rewrite rule inevitably involves sub-graphs related by $\lhd^{\vec{@}}$, and as a result, can yield different results, i.e. size($|\dot{\mathcal{C}}[\vec{N}]|$) vs. size($|\dot{\mathcal{C}}[\vec{H}]|$). The same rewrite rule can be applied to states $\dot{\mathcal{C}}[\vec{N}], \dot{\mathcal{C}}[\vec{H}]$ and yield $\dot{P}, \dot{P}'$, respectively, and these states $\dot{P}, \dot{P}'$ can contain different natural-number edges (i.e. n vs. m such that $n \geq m$). We cannot take a specimen of $\lhd^{\vec{@}}$ to make (8.1) happen. ■

We can therefore observe the following: the micro-beta template $\lhd^{\vec{@}}$ satisfies (8.1) relative to active operations $\{+, -, \vec{@}\} = \mathbb{O}^{\text{lin}}_{\lightning} \backslash \{\texttt{stat}\}$ (formalised in Lemma 8.9 below), and does not satisfy (8.1) relative to the active operation `stat`. We say the micro-beta template $\lhd^{\vec{@}}$ is *robust* only relative to $\mathbb{O}^{\text{lin}}_{\lightning} \backslash \{\texttt{stat}\}$. Robustness, as a sufficient condition for (8.1), can be formalised as follows.

Definition 8.8 (Robustness)**.** A pre-template $\lhd$ is $(\mathbb{C}, Q, Q', Q'')$-*robust relative to* a rewrite transition $\dot{N} \to \dot{N}'$ if, for any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}; \vec{H^1}; \vec{H^2})$ of $\lhd$, such that $\dot{\mathcal{C}}[\vec{H^1}] = \dot{N}$ and the focus of $\dot{\mathcal{C}}$ is the $\lightning$-focus and not entering, one of the following holds.

(I) $\dot{\mathcal{C}}[\vec{H^1}]$ or $\dot{\mathcal{C}}[\vec{H^2}]$ is not rooted.

(II) There exists a stuck state $\dot{N}''$ such that $\dot{N}' \to^* \dot{N}''$.

(III) There exist a quasi-$\mathbb{C}$-specimen $(\dot{N}''_1, \dot{N}''_2)$ of $\lhd$ up to $(\dot{\preceq}_{Q'}, \dot{\preceq}_{Q''})$, whose focus is not the $\lightning$-focus, and two numbers $k_1, k_2 \in \mathbb{N}$, such that $(1 + k_1)\ Q\ k_2$, $\dot{N}' \to^{k_1} \dot{N}''_1$, and $\dot{\mathcal{C}}[\vec{H^2}] \to^{k_2} \dot{N}''_2$.

We can now formally say that the micro-beta pre-template $\vec{\lhd}^{@}$ is robust relative to the substitution transitions and the extrinsic transitions for $\mathbb{O}^{\mathrm{lin}}\backslash\{\mathtt{stat}\}$. The third and fourth parameters of robustness are irrelevant here[5], and we can simply set them as the equality $=$.

Lemma 8.9 (Robustness of $\vec{\lhd}^{@}$)**.** *The micro-beta pre-template $\vec{\lhd}^{@}$ is $(\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}\backslash\{\mathtt{stat}\}}, =, =, =)$-robust relative to the substitution transitions and the extrinsic transitions for $\mathbb{O}^{\mathrm{lin}}_{\lightning}\backslash\{\mathtt{stat}\}$.* □

8.4. **A sufficiency-of-robustness theorem.** We have collected definitions of safety and robustness. We can finally state the sufficiency-of-robustness theorem. The theorem incorporates the so-called *up-to* technique: it enables us to prove contextual refinement $\preceq^{\mathbb{C}}_{Q}$ that depends on (or, that is *up to*) state refinements $\dot{\preceq}_{Q'}$ and $\dot{\preceq}_{Q''}$. Safety (Definition 8.4) and robustness (Definition 8.8) accordingly incorporates the up-to technique by means of quasi-specimens up-to. We cannot use arbitrary preorders Q', Q'' in combination with the preorder Q, though; they must be *reasonable* in the following sense.

Definition 8.10 (Reasonable triples)**.** A triple (Q, Q', Q'') of preorders on $\mathbb{N}$ is *reasonable* if the following hold:

(A) Q is closed under addition, i.e. any $k_1\ Q\ k_2$ and $k'_1\ Q\ k'_2$ satisfy $(k_1 + k'_1)\ Q\ (k_2 + k'_2)$.

(B) $Q' \subseteq \geq_{\mathbb{N}}$, and $Q' \subseteq Q''$.

(C) $Q' \circ Q \circ Q'' \subseteq Q$, where $\circ$ denotes composition of binary relations.

Examples of a reasonable triple (Q, Q', Q'') include: $(\mathbb{N}\times\mathbb{N}, \geq_{\mathbb{N}}, \mathbb{N}\times\mathbb{N})$, $(\mathbb{N}\times\mathbb{N}, \geq_{\mathbb{N}}, \geq_{\mathbb{N}})$, $(\mathbb{N}\times\mathbb{N}, =_{\mathbb{N}}, =_{\mathbb{N}})$, $(\geq_{\mathbb{N}}, \geq_{\mathbb{N}}, \geq_{\mathbb{N}})$, $(\geq_{\mathbb{N}}, =_{\mathbb{N}}, \geq_{\mathbb{N}})$, $(\leq_{\mathbb{N}}, =_{\mathbb{N}}, \leq_{\mathbb{N}})$, $(\geq_{\mathbb{N}}, =_{\mathbb{N}}, =_{\mathbb{N}})$, $(\leq_{\mathbb{N}}, =_{\mathbb{N}}, =_{\mathbb{N}})$, $(=_{\mathbb{N}}, =_{\mathbb{N}}, =_{\mathbb{N}})$.

The sufficiency-of-robustness theorem has two clauses (1) and (2). To prove that a pre-template $\lhd$ implies contextual equivalence (not refinement), one can use (1) twice with respect to $\lhd$ and $\lhd^{-1}$, or alternatively use (1) and (2) with respect to $\lhd$. The alternative approach is often more economical. This is because it involves proving input-safety of $\lhd$ for both parameters Q and Q^{-1}, which typically boils down to a single proof for the smaller one of Q, Q^{-1}, thanks to monotonicity of input-safety with respect to Q.

Theorem 8.11 (Sufficiency-of-robustness theorem)**.** *If an universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is deterministic and refocusing, it satisfies the following property. For any set $\mathbb{C} \subseteq \mathcal{H}_{\omega}(L_{\mathrm{lin}}, M_{\mathrm{lin}} \cup \mathbb{M})$ of contexts that is closed under plugging, any reasonable triple (Q, Q', Q''), and any pre-template $\lhd$ on focus-free hypernets $\mathcal{H}_{\omega}(L_{\mathrm{lin}}, M_{\mathrm{lin}}(\mathbb{O})\backslash\{?, \checkmark, \lightning\})$:*

(1) *If $\lhd$ is a $(\mathbb{C}, Q, Q')$-template and $(\mathbb{C}, Q, Q', Q'')$-robust relative to all rewrite transitions, then $\lhd$ implies contextual refinement in $\mathbb{C}$ up to Q, i.e. any $G_1 \lhd G_2$ implies $G_1 \preceq^{\mathbb{C}}_{Q} G_2$.*

(2) *If $\lhd$ is a $(\mathbb{C}, Q^{-1}, Q')$-template and the converse $\lhd^{-1}$ is $(\mathbb{C}, Q, Q', Q'')$-robust relative to all rewrite transitions, then $\lhd^{-1}$ implies contextual refinement in $\mathbb{C}$ up to Q, i.e. any $G_1 \lhd G_2$ implies $G_2 \preceq^{\mathbb{C}}_{Q} G_1$.*

Proof. This is a consequence of Proposition F.6, Proposition F.2 and Proposition F.4 in Appendix F.

[5]These parameters will be used in Section 12 (See Table 3).

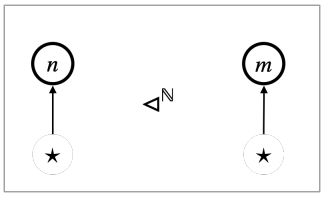


Figure 12: An auxiliary pre-template, where $n, m \in \mathbb{N}$

The proof is centred around a notion of *counting simulation* (Definition F.1). We first prove that, for each robust template $\lhd$, its contextual closure $\overline{\lhd}$ is a counting simulation (Proposition F.6). We then prove soundness of counting simulation with respect to state refinement (Proposition F.2). We can further prove that, if $\overline{\lhd}$ implies state refinement, $\lhd$ implies contextual refinement (Proposition F.4). □

We can use Theorem 8.11 (1) to conclude that the micro-beta pre-template implies contextual refinement $\preceq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}\setminus\{\mathtt{stat}\}}}_{\geq\mathbb{N}}$. Note that The micro-beta pre-template $\lhd^{\vec{@}}$ is $(\mathbb{C}_{\mathbb{O}}, =, =, =)$-robust, and therefore $(\mathbb{C}_{\mathbb{O}}, \geq, =, =)$-robust as well, because $= \subseteq \geq$.

Proposition 8.12. *For the operation set $\mathbb{O}^{\mathrm{lin}}\setminus\{\mathtt{stat}\}$, the micro-beta pre-template $\lhd^{\vec{@}}$ implies contextual refinement $\preceq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}\setminus\{\mathtt{stat}\}}}_{\geq\mathbb{N}}$.* □

The above proposition establishes contextual refinement in the absence of the operation `stat`. This is due to the observation that the micro-beta template $\lhd^{\vec{@}}$ is not robust relative to the rewrite rule of `stat` (Figure 10c). Can we say more than just failure of robustness, in the presence of `stat`?

The answer is yes. For the specific operation set $\mathbb{O}^{\mathrm{lin}}$, we can actually show that the micro-beta template $\lhd^{\vec{@}}$ implies contextual refinement $\preceq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}}_{\geq\mathbb{N}}$.

Lemma 8.13 (Robustness of $\lhd^{\vec{@}}$)**.** *The micro-beta pre-template $\lhd^{\vec{@}}$ is $(\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}, =, =, =)$-robust relative to the rewrite transitions for* `stat`*.*

Proof. To prove robustness, we use the auxiliary pre-template $\lhd^{\mathbb{N}}$, shown in Figure 12, that identifies all natural number edges. The pre-template is a $(\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}, =, =)$-template, and it is $(\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}, =, =, =)$-robust relative to the substitution transitions and the extrinsic transitions for $\mathbb{O}^{\mathrm{lin}}_{\ell}$ including `stat`. Therefore, by Theorem 8.11 (1), it implies contextual refinement $\preceq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}}_{=\mathbb{N}}$. The pre-template is symmetric, and consequently, it implies contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}}_{=\mathbb{N}}$.

In Case (iii-4) in Section 8.3, we could not take a specimen of $\lhd^{\vec{@}}$ that induces the states $\dot{P}, \dot{P}'$. The difference between these states is given by $\lhd^{\vec{@}}$ as well as different natural number edges; in other words, the difference is given by $\lhd^{\vec{@}}$ and $\lhd^{\mathbb{N}}$.

In fact, we *can* take a *quasi*-specimen of $\lhd^{\vec{@}}$ up to $(\doteq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}}_{=\mathbb{N}}, \doteq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}}_{=\mathbb{N}})$, which is namely the pair $(\dot{P}, \dot{P}')$. There exists a state $\dot{P}''$ such that: (a) $\dot{P} \doteq^{\mathbb{C}_{\mathbb{O}^{\mathrm{lin}}}}_{=\mathbb{N}} \dot{P}''$ thanks to the symmetric robust template $\lhd^{\mathbb{N}}$, and (b) the pair $(\dot{P}'', \dot{P}')$ is induced by a specimen of $\lhd^{\vec{@}}$.

$$
\begin{array}{lll}
\text{Constants:} & c ::= n & \text{(natural numbers)} \\
& \quad \mid \mathtt{tt} \mid \mathtt{ff} & \text{(booleans)} \\
& \quad \mid () & \text{(unit value)} \\
\text{Unary operations:} & \$_1 ::= -_1 & \text{(negation of natural numbers)} \\
& \quad \mid \mathtt{ref} & \text{(reference creation)} \\
& \quad \mid ! & \text{(dereferencing)} \\
\text{Binary operations:} & \$_2 ::= + \mid - & \text{(summation/subtraction of natural numbers)} \\
& \quad \mid := & \text{(assignment)} \\
& \quad \mid = & \text{(equality testing of atoms)} \\
\text{Terms:} & t, u ::= x \mid \lambda x.\, t \mid t\ u & \text{(the lambda-calculus terms)} \\
& \quad \mid a & \text{(atoms)} \\
& \quad \mid c \mid \$_1\, t \mid t\, \$_2\, u & \text{(constants, unary/binary operations)}
\end{array}
$$

Formation rules:

$$\frac{}{\Gamma_1, x, \Gamma_2 \mid \Delta \vdash x} \qquad \frac{}{\Gamma \mid \Delta_1, a, \Delta_2 \vdash a}$$

$$\frac{x, \Gamma \mid \Delta \vdash t}{\Gamma \mid \Delta \vdash \lambda x.\, t} \qquad \frac{\Gamma \mid \Delta \vdash t \quad \Gamma \mid \Delta \vdash u}{\Gamma \mid \Delta \vdash t\ u} \qquad \frac{}{\Gamma \mid \Delta \vdash c}$$

$$\frac{\Gamma \mid \Delta \vdash t}{\Gamma \mid \Delta \vdash \$_1\, t} \qquad \frac{\Gamma \mid \Delta \vdash t \quad \Gamma \mid \Delta \vdash u}{\Gamma \mid \Delta \vdash t\, \$_2\, u}$$

Syntactic sugar:

$$\mathtt{let}\ x = u\ \mathtt{in}\ t \stackrel{def}{=} (\lambda x.\, t)\ u$$

$$u; t \stackrel{def}{=} (\lambda z.\, t)\ u \qquad \text{(where } z \text{ is a fresh variable)}$$

Figure 13: An extended call-by-value lambda-calculus with variable sharing and store

As a result, with the help of the symmetric robust template $\lhd^{\mathbb{N}}$, the micro-beta template $\lhd^{\vec{@}}$ is robust also relative to `stat`. □

In the presence of conditional branching and divergence, however, the micro-beta template $\lhd^{\vec{@}}$ would not be robust relative to `stat`. The above lemma crucially relies on robustness of the auxiliary template $\lhd^{\mathbb{N}}$, which would be violated by conditional branching; in other words, the template $\lhd^{\mathbb{N}}$ would not be robust relative to conditional branching. Conditional branching and divergence, combined with `stat`, would enable us to construct a specific context that distinguishes the left- and the right hand sides of $\lhd^{\vec{@}}$.

9. Representation of variable sharing and store

This section adapts the translation $(-)^\dagger$ of the linear lambda-calculus, presented in Section 4, to accommodate variable sharing and (general, untyped) store. Figure 13 shows an extension of the untyped call-by-value lambda-calculus that accommodates arithmetic as well as:

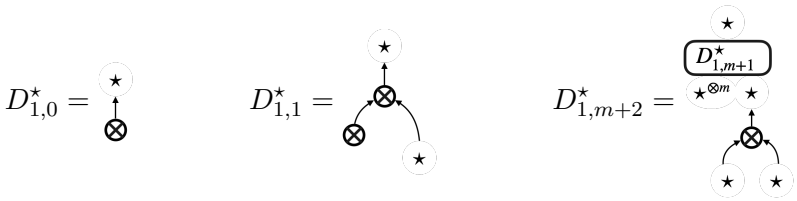
- atoms (or, locations of store) a,

- reference-related operations: reference creation `ref`, dereferencing !, assignment :=, equality testing of atoms =,
- and their return values: booleans `tt`, `ff` and the unit value ().

In Figure 13, Γ is a finite sequence of (free) variables and Δ is a finite sequence of (free) atoms.

9.1. **Variable sharing.** In an arbitrary term, a variable may occur many times, or it may not occur at all. To let sub-terms share and discard a variable, we introduce *contraction* edges $\otimes^\star_{\mathsf{C}} \colon \star^{\otimes 2} \Rightarrow \star$ and *weakening* edges $\otimes^\star_{\mathsf{W}} \colon \epsilon \Rightarrow \star$. We can construct a binary tree with an arbitrary number of leaves using these contraction and weakening edges. We call such a tree *contraction tree*.

There are many ways to construct a contraction tree of m leaves, but we fix a *canonical*[6] tree $D^\star_{1,m} \colon \star^{\otimes m} \Rightarrow \star$ as below.



Every canonical tree $D^\star_{1,m}$ contains exactly one weakening edge and m contraction edges.

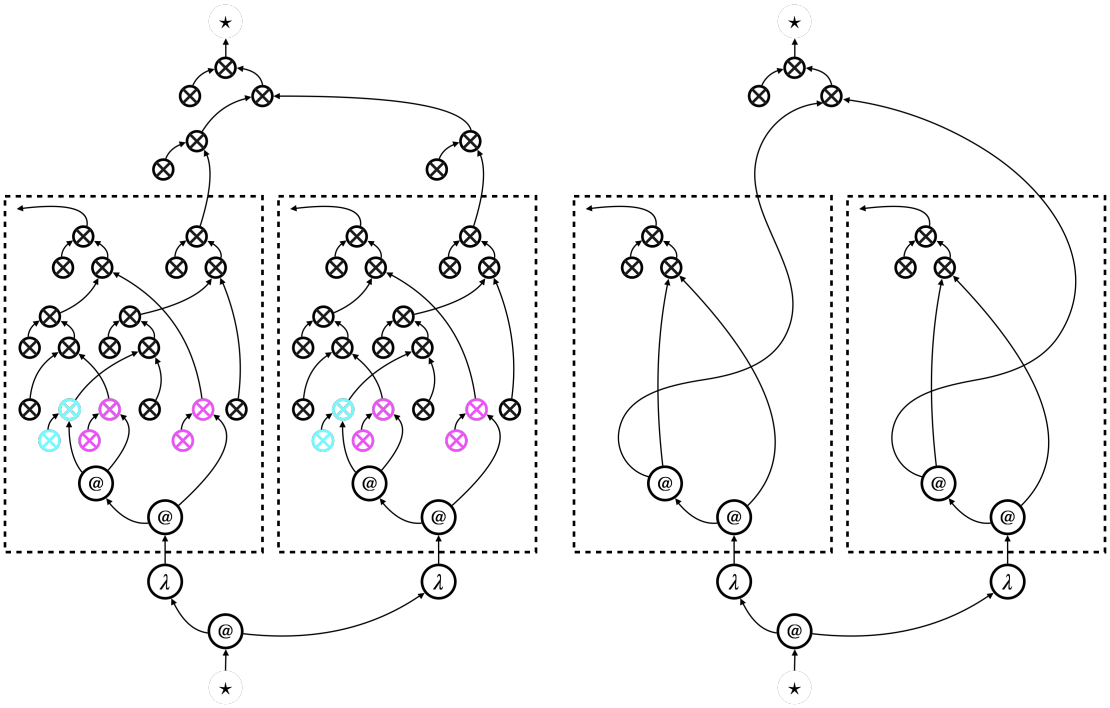
The translation $(-)^\dagger$ of linear lambda-terms needs to be adapted to yield a translation $(-)^\ddagger$ of general lambda-terms. The adaptation amounts to inserting canonical trees appropriately. Additionally, we replace the edges $\mathsf{V} \colon \star \Rightarrow \star$ with canonical trees $D^\star_{1,1}$, to obtain uniform translation.

We will present the exact translation $(-)^\ddagger$ in Section 9.3 (see Figure 16), and here we just show an example in Figure 14. Figure 14a shows the hypernet $\big(f \vdash (\lambda x.f\ x\ x)\ (\lambda x.f\ x\ x)\big)^\ddagger \colon \star \Rightarrow \star$. Canonical trees $D^\star_{1,0}, D^\star_{1,1}, D^\star_{1,2}$ are inserted for each term constructor, but among those, coloured canonical trees $D^\star_{1,1}$ replace the edges $\mathsf{V} \colon \star \Rightarrow \star$ that would represent variable occurrences in the linear translation $(-)^\dagger$. Figure 14b shows a possible simplification of the hypernet; such simplification is validated only by proving certain observational equivalences on contraction trees (namely, those in Figure 21).

9.2. **Store and atom sharing.** We next look at terms that contain references to store. An example term is $a := (!a + 1)$ that increments the value stored at atom (location) a. We represent such a term, together with store (e.g. $\{a \mapsto 0\}$), altogether as a single hypernet; see Figure 15. The hypernets in the figure are simplified in the same manner as Figure 14b.

To represent store and atom sharing, we begin with an observation about the difference between variable sharing and atom sharing. Atoms, like variables, may occur arbitrarily many times in a single term. However, whereas terms bound to a variable can be duplicated, each store associated to the atom should not be duplicated. For example, the atom a appears twice in the term $a := (!a + 1)$, but this does not mean the store $\{a \mapsto 0\}$ is accordingly duplicated. Both occurrences of the atom a points at the single store $\{a \mapsto 0\}$.

[6]Each $D^\star_{1,m}$ is canonical, in the sense that any contraction tree that contains at least one weakening and has m leaves can be simplified to $D^\star_{1,m}$ using the laws in Figure 21 (namely $\lhd^{\vec{@}\,\text{Assoc}}, \lhd^{\vec{@}\,\text{Comm}}, \lhd^{\vec{@}\,\text{Idem}}$).



(a) Original (b) Simplified

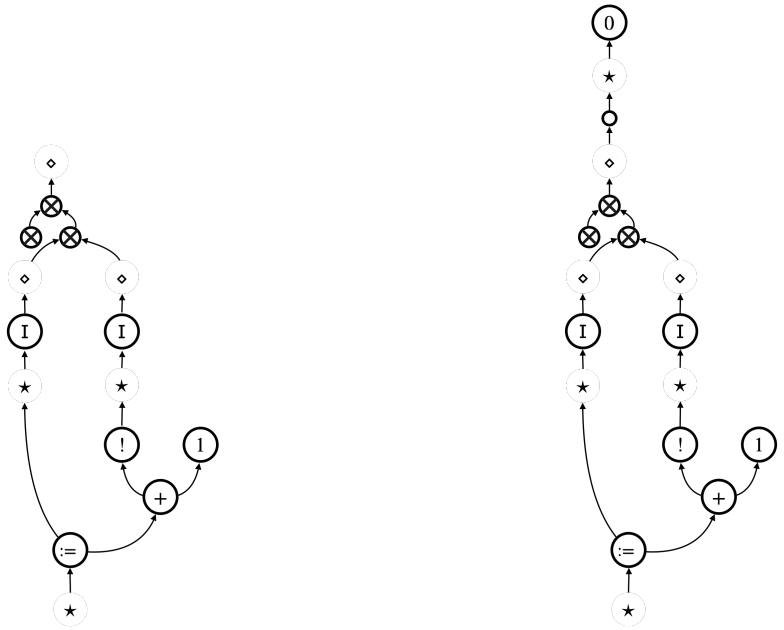
Figure 14: The hypernet $\left(f \vdash (\lambda x.f\ x\ x)\ (\lambda x.f\ x\ x)\right)^{\ddagger} \colon \star \Rightarrow \star$ and its simplification

This observation leads us to introduce one new vertex label, and some edge labels, of our hypernets. Firstly, we introduce a new vertex label $\diamond$ that represents store, in addition to the labels we have used (i.e. $\star$ for terms and $T^n(\star)$ for thunks with n bound variables). While terms and thunks can both be duplicated, store cannot be duplicated; this is the reason why we need a new vertex label.

Secondly, we introduce another version of contraction and weakening, namely $\otimes^{\diamond}_{\mathsf{C}} \colon \diamond^{\otimes 2} \Rightarrow \diamond$ and $\otimes^{\diamond}_{\mathsf{W}} \colon \epsilon \Rightarrow \diamond$, for the type $\diamond$. We call these contraction and weakening as well. Contractions and weakenings for $\star$ and $\diamond$ appear the same, but they will have different behaviours (see Section 10.1).

Finally, we introduce edge labels for type conversion between $\star$ and $\diamond$: namely, anonymous atom edges called *instance* $\mathsf{I} \colon \star \Rightarrow \diamond$, and anonymous store edges $\circ \colon \diamond \Rightarrow \star$. Their roles in the hypernet representation are best understood looking at the example in Figure 15. Each occurrence of an atom is represented by the anonymous instance edge 'I', and instances of the same atom is connected to a contraction tree of type $\diamond$. The contraction tree is then connected to the part of the hypernet in Figure 15b that represents the store $\{a \mapsto 0\}$, which consists of the store edge '$\circ$' and the stored value 0. In Figure 15b, the part of the hypernet that is typed by $\diamond$ connects instances of an atom to its stored value.

To represent sharing of (instances of) atoms, we will use *canonical* contraction trees $D^{\diamond}_{1,m}$ that are constructed in the same manner as the canonical trees $D^{\star}_{1,m}$. Every canonical tree $D^{\diamond}_{1,m}$ again contains exactly one weakening edge and m contraction edges.



(a) The term alone (b) With store

Figure 15: The hypernet $(a \vdash a := (!a + 1))^{\ddagger} \colon \star \Rightarrow \diamond$ with store $\{a \mapsto 0\}$, simplified

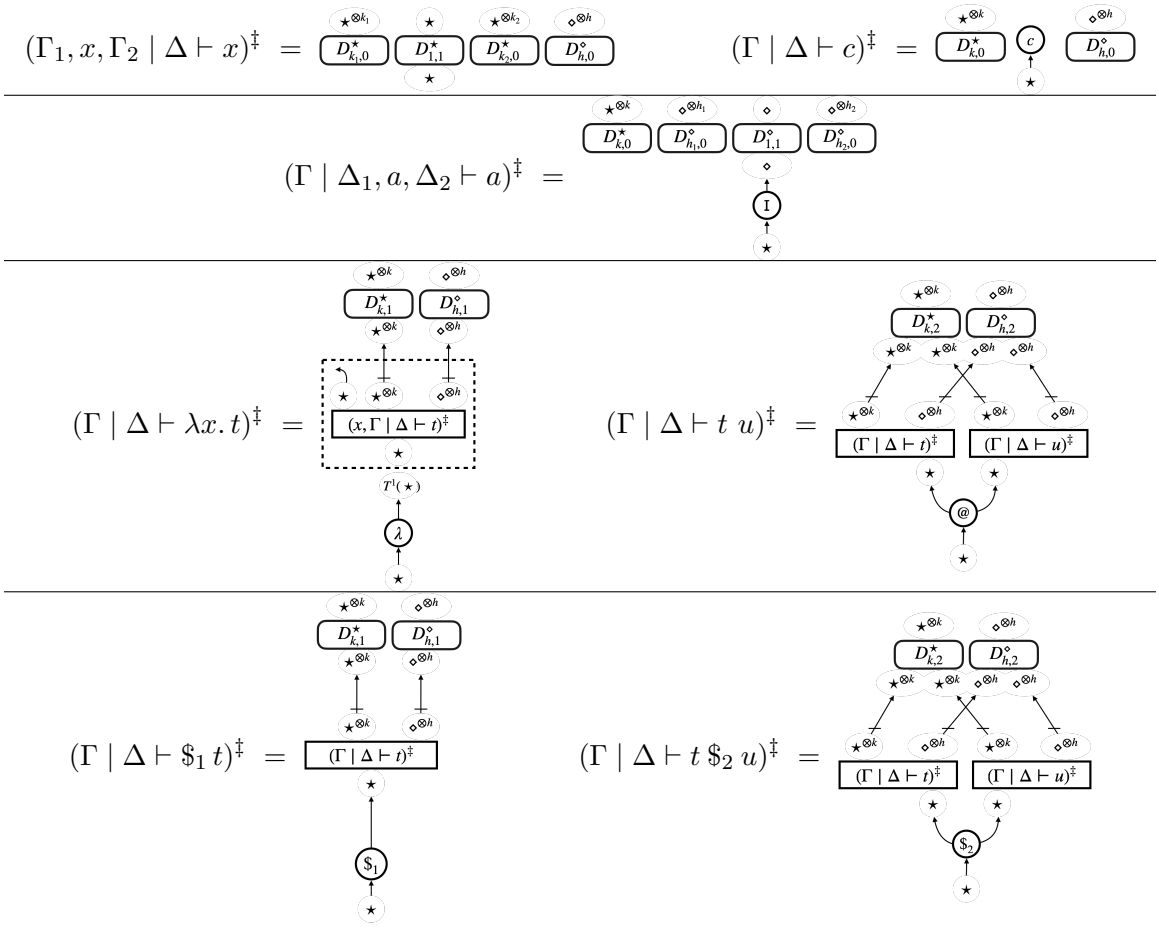
9.3. **Inductive translation** $(-)^{\ddagger}$**.** We now present the translation $(-)^{\ddagger}$ of the extended lambda-terms into hypernets $\mathcal{H}_{\omega}(L_{\text{gen}}, M_{\text{gen}})$ where

$$L_{\text{gen}} = \{\star, \diamond\} \cup \{T^n(\star) \mid n \in \mathbb{N}\}, \tag{9.1}$$

$$\begin{aligned} M_{\text{gen}} = {} & \{\lambda \colon \star \Rightarrow T^1(\star),\ \overrightarrow{@} \colon \star \Rightarrow \star^{\otimes 2},\ + \colon \star \Rightarrow \star^{\otimes 2},\ -_2 \colon \star \Rightarrow \star^{\otimes 2},\ -_1 \colon \star \Rightarrow \star,\} \\ & \cup \{\mathtt{ref} \colon \star \Rightarrow \star,\ ! \colon \star \Rightarrow \star,\ = \colon \star \Rightarrow \star^{\otimes 2},\ := \colon \star \Rightarrow \star^{\otimes 2}\} \\ & \cup \{\mathtt{tt} \colon \star \Rightarrow \epsilon,\ \mathtt{ff} \colon \star \Rightarrow \epsilon,\ () \colon \star \Rightarrow \epsilon\} \\ & \cup \{n \colon \star \Rightarrow \epsilon \mid n \in \mathbb{N}\} \\ & \cup \{\otimes^{\star}_{\mathsf{C}} \colon \star^{\otimes 2} \Rightarrow \star,\ \otimes^{\star}_{\mathsf{W}} \colon \epsilon \Rightarrow \star,\ \otimes^{\diamond}_{\mathsf{C}} \colon \diamond^{\otimes 2} \Rightarrow \diamond,\ \otimes^{\diamond}_{\mathsf{W}} \colon \epsilon \Rightarrow \diamond\}. \end{aligned} \tag{9.2}$$

In general, a judgement $\Gamma \mid \Delta \vdash t$ is translated into $t^{\ddagger} \colon \star \Rightarrow \star^{\otimes k} \otimes \diamond^{\otimes h}$ where the lengths of Γ, Δ are k, h respectively.

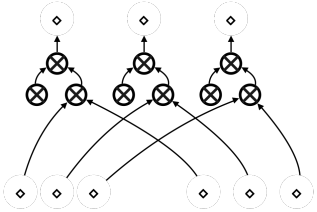
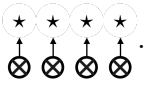
The translation $(-)^{\ddagger}$ uses a generalisation of canonical trees $D^{\ell}_{1,m} \colon \ell^{\otimes m} \Rightarrow \ell$ (where $\ell \in \{\star, \diamond\}$), namely a forest $D^{\ell}_{k,m}$ of canonical trees dubbed *distributor*. A distributor $D^{\ell}_{k,m} \colon \ell^{\otimes km} \Rightarrow \ell^{\otimes k}$ is a forest of k canonical trees $D^{\ell}_{1,m}$ with some permutation of inputs. It is inserted in the translation for sharing k variables/atoms (depending on $\ell \in \{\star, \diamond\}$) among



where k, k_1, k_2, h, h_1, h_2 are the lengths of $\Gamma, \Gamma_1, \Gamma_2, \Delta, \Delta_1, \Delta_2$ respectively.

Figure 16: Inductive translation $(-)^{\ddagger}$ of well-formed lambda-terms

m sub-terms. For instance,

$D^{\diamond}_{3,2} =$  and $D^{\star}_{4,0} =$  .

The precise definition of distributors will be given in Section 10.2.

10. The copying UAM

10.1. **New behaviour.** We equip the UAM with *copying*, which is the behaviour of contraction edges $\otimes^{\star}_{\mathsf{C}} \colon \star^{\otimes 2} \Rightarrow \star$. The copying UAM $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ acts on hypernets $\mathcal{H}_{\omega}(L_{\mathrm{gen}}, M_{\mathrm{gen}}(\mathbb{O}))$

<table>
<tr><th colspan="2">transitions</th><th>focus</th><th>provenance</th></tr>
<tr><td colspan="2">search transitions</td><td>?, ✓</td><td rowspan="2">intrinsic</td></tr>
<tr><td rowspan="2">rewrite transitions</td><td>copy transitions</td><td rowspan="2">↯</td></tr>
<tr><td>behaviour $B_{\mathbb{O}}$
(compute transitions)</td><td>extrinsic</td></tr>
</table>

Table 2: Transitions of the copying UAM

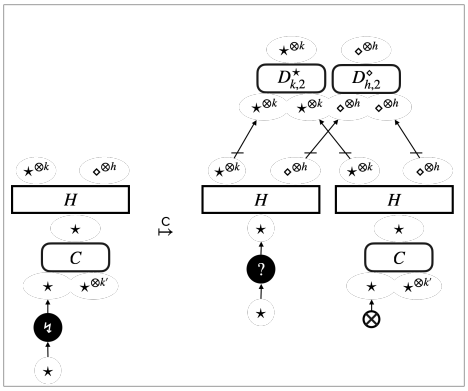
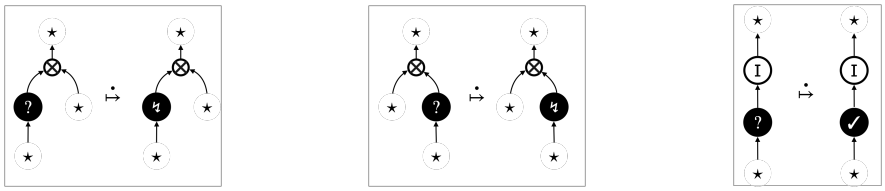


Figure 17: The copy rule where H is a copyable hypernet



(a) (b) (c)

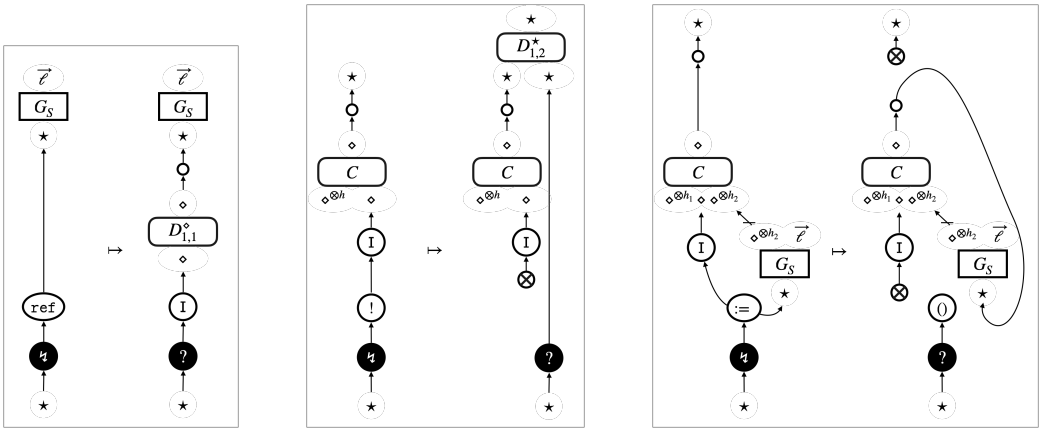
Figure 18: Extra interaction rules

where

$$M_{\text{gen}}(\mathbb{O}) = \mathbb{O} \cup \{?\colon \star \Rightarrow \star,\ \checkmark\colon \star \Rightarrow \star,\ \lightning\colon \star \Rightarrow \star\} \tag{10.1}$$
$$\cup\, \{\otimes^{\star}_{\mathsf{C}}\colon \star^{\otimes 2} \Rightarrow \star,\ \otimes^{\star}_{\mathsf{W}}\colon \epsilon \Rightarrow \star,\ \otimes^{\diamond}_{\mathsf{C}}\colon \diamond^{\otimes 2} \Rightarrow \diamond,\ \otimes^{\diamond}_{\mathsf{W}}\colon \epsilon \Rightarrow \diamond\}.$$

For box edges, we impose the following type discipline: each box edge must have a type $T^n(\star) \Rightarrow \star^{\otimes m} \otimes \diamond^{\otimes h}$ with its content having a type $\star \Rightarrow \star^{\otimes(n+m)} \otimes \diamond^{\otimes h}$.

Table 2 summarises transitions of the copying UAM. The copying UAM has *copy* transitions instead of substitution transitions (cf. Table 1). Copy transitions are specified by the copy rule shown in Figure 17. The copy rule duplicates a *copyable* hypernet, and inserts distributors and a weakening edge. A copyable hypernet consists of instance edges, operation edges, and box edges. The precise definition of copyable hypernets will be given in Section 10.2. Note that the copy rule only applies to contraction of type $\star$; contraction of type $\diamond$, which are for atoms, do not have a corresponding copy rule. This reflects the fact that atoms are never duplicated unless it is inside a thunk.



(a) Reference creation (b) Dereferencing (c) Assignment

(d) Equality testing

Figure 19: The example behaviour $B_{\{\texttt{ref},!,:=,=\}}$ where C, C' are contraction trees and G_S is a hypernet

Accordingly, the copying UAM has extra search transitions concerning contractions ('$\otimes$') and instances ('I'). These are specified by the interaction rules shown in Figure 18.

Figure 19 shows an example of the behaviour $B_{\mathbb{O}}$, namely that for the following active operations for store:

$$\texttt{ref} : \star \Rightarrow \star, \quad ! : \star \Rightarrow \star, \quad := : \star \Rightarrow \star^{\otimes 2}, \quad = : \star \Rightarrow \star^{\otimes 2}.$$

Their behaviour is specified locally by rewrite rules.

Figure 19a: This rewrite rule specifies the behaviour of reference creation. The rule introduces store ('$\circ$') and its instance ('I'). In between these two edges, a canonical tree $D^{\diamond}_{1,1}$ is inserted so that there is always a contraction tree between any store and instances.

Figure 19b: This rewrite rule specifies the behaviour of dereferencing. When dereferencing is triggered, the store edge ('$\circ$') gets attached to a canonical tree $D^{\star}_{1,2}$, so that its stored value will be copied by subsequent transitions.

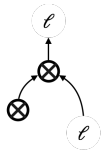
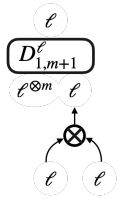
Figure 19c: This rewrite rule specifies the behaviour of assignment. It identifies the store edge ('$\circ$') by tracing the contraction tree C connected to the instance edge ('I'). Once

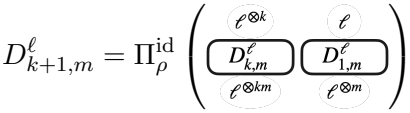
the store edge is identified, the current stored value gets disconnected and the new value G_S gets connected to the store edge. Note that the value G_S may contain other instances referring to the same store; this means the rewrite rule may introduce a cycle (via $\diamond^{\otimes h_2}$).

Figure 19d: This rewrite rule specifies the behaviour of equality testing for atoms. It determines whether two instance edges ('I') are connected to the same store edge ('○') or not, by tracing contraction trees C, C'. If they are connected to the same store, the result `tt` is introduced and attached to the ?-focus. If not, the result `ff` is introduced instead.

10.2. **Auxiliary definitions.** We here present a few definitions of the concepts introduced so far. The first is *distributors*. A distributor $D^{\ell}_{k,m}$ ($\ell \in \{\star, \diamond\}$) is given by a forest of canonical trees $D^{\ell}_{1,m}$ whose inputs are permuted. The permutation is realised by *interface permutation* (Definition 3.5).

Definition 10.1 (Distributors and canonical trees)**.** For $\ell \in \{\star, \diamond\}$ and $k, m \in \mathbb{N}$, a *distributor* is defined inductively as follows.

$D^{\ell}_{0,m} = \emptyset$ $\quad$ $D^{\ell}_{1,0} =$  $\quad$ $D^{\ell}_{1,1} =$  $\quad$ $D^{\ell}_{1,m+2} =$ 

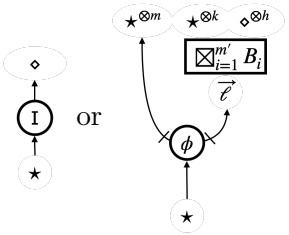
$$D^{\ell}_{k+1,m} = \Pi^{\mathrm{id}}_{\rho}$$ 

where $\emptyset$ denotes the empty hypernet, id is the identity map, and ρ is a bijection such that, for each $j \in \{1, \ldots, k\}$ and $i \in \{1, \ldots, m\}$, $\rho(j + (k+1)(i-1)) = j + k(i-1)$ and $\rho((k+1)i) = km + i$.

When $k = 1$, the distributor $D^{\ell}_{1,m}$ is called *canonical tree*.

The second is *copyable* hypernets. The copy rule (Figure 17) duplicates a single copyable hypernet at a time.

Definition 10.2 (Copyable hypernets)**.** A hypernet $H : \star \Rightarrow \star^{\otimes k} \otimes \diamond^{\otimes h}$ is called *copyable* if it is given by



where $\phi \in \mathbb{O}$ and $\boxtimes_{i=1}^{m'} B_i$ is a (possibly empty) parallel juxtaposition of box hypernets.

Finally, the notion of *stable hypernet* (Definition 6.5), which corresponds to values, needs to change to incorporate store.

Definition 10.3 (Stable hypernets)**.** A *stable* hypernet is a hypernet $(G : \star \Rightarrow \otimes_{i=1}^{m} \ell_i) \in \mathcal{H}(L_{\mathrm{gen}}, \{\mathsf{I}\} \cup \mathbb{O}_{\checkmark})$, such that $\otimes_{i=1}^{m} \ell_i \in (\{\diamond\} \cup \{T^n(\star) \mid n \in \mathbb{N}\})^m$ and each vertex is reachable from the unique input.

10.3. **Determinism and refocusing.** The properties of determinism and refocusing can be defined for the copying UAM in the same way as the original UAM (cf. Definition 7.3). We can again use the stationary property as a sufficient condition for refocusing (Lemma 7.5). Determinism and refocusing of a copying UAM also boil down to those of extrinsic transitions $B_{\mathbb{O}}$ under a mild condition.

Lemma 10.4 (Determinism and refocusing)**.**

- *A copying universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is deterministic if extrinsic transitions $B_{\mathbb{O}}$ are deterministic.*
- *Suppose that G_S in the substitution rule (Figure 9f) is a stable hypernet. A copying universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is refocusing if extrinsic transitions $B_{\mathbb{O}}$ preserve the rooted property.*

Proof. The proof is that of Lemma 7.6 equipped with analysis of copying transitions. Copy transitions are all deterministic, because different contraction rules applied to a single state result in the same state. The choice of the contraction tree (C in Figure 17) is irrelevant. Copy transitions are stationary, and hence they preserve the rooted property. □

11. Observational equivalence on lambda-terms

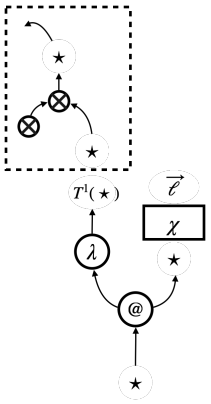
Let $\mathbb{O}^{\mathrm{ex}} = \mathbb{O}^{\mathrm{ex}}_{\checkmark} \uplus \mathbb{O}^{\mathrm{ex}}_{\not\downarrow}$ be the set of operations given by $\mathbb{O}^{\mathrm{ex}}_{\checkmark} = \mathbb{N} \cup \{\lambda, \mathtt{tt}, \mathtt{ff}, ()\}$ and $\mathbb{O}^{\mathrm{ex}}_{\not\downarrow} = \{+, -, -_1, \vec{@}, \mathtt{ref}, !, :=, =\}$. The copying UAM $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B^{\mathrm{ex}}_{\mathbb{O}})$ provides operational semantics of the extended lambda-calculus (Figure 13). Using contextual equivalence on hypernets (Definition 7.2), we can define a notion of observational equivalence on lambda-terms. The notion only concerns the coincidence of termination, which is standard given that the extended lambda-calculus is untyped.

Definition 11.1 (Observational equivalence on lambda-terms)**.** Let $\Gamma \mid \Delta \vdash t$ and $\Gamma \mid \Delta \vdash u$ be two derivable judgements. The lambda-terms t and u are said to be *observationally equivalent*, written as $\Gamma \mid \Delta \vDash t \simeq^{\ddagger} u$, if $(\Gamma \mid \Delta \vdash t)^{\ddagger} \simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}}_{\mathbb{N}\times\mathbb{N}} (\Gamma \mid \Delta \vdash t)^{\ddagger}$ holds.

This definition uses the specific contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}}_{\mathbb{N}\times\mathbb{N}}$. Firstly, the use of the universal relation $\mathbb{N} \times \mathbb{N}$ makes the number of transitions until termination irrelevant. Secondly, the set $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$ is the set of all *binding-free* contexts.

Definition 11.2 (Binding-free contexts)**.** A focus-free context $\mathcal{C}$ is said to be *binding-free* if there exists no path, at any depth, from a source of a contraction, atom, box or hole edge, to a source of a hole edge.

For example,



is a binding-free context where χ is a hole. Syntactically it would coincide with a context $(\lambda x.x)\ \Box$.

Note that the contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}}_{\mathbb{N}\times\mathbb{N}}$ is a larger relation than the contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{\mathbb{N}\times\mathbb{N}}$ with respect to the set $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$ of all contexts.

The restriction to binding-free contexts can be justified by the fact that the observational equivalence $\simeq^{\ddagger}$ is a congruence relation with respect to lambda-contexts defined by the following grammar:

$$C ::= [\,] \mid \lambda x.\, C \mid C\ t \mid t\ C \mid \$_1\, C \mid C\, \$_2\, t \mid t\, \$_2\, C$$

where $\$_1 \in \{\mathtt{ref}, !, -_1\}$ and $\$_2 \in \{=, :=, +, -\}$. This congruence property will be formalised as Lemma 11.3 below.

We can extend the translation $(-)^{\ddagger}$ to accommodate lambda-contexts. To do so, we first adapt the formation rules of lambda-terms in Figure 13 to lambda-contexts, by annotating the hole '$[\,]$' as '$[\,]_{\Gamma|\Delta}$' and adding a formation rule $\overline{\Gamma \mid \Delta \vdash [\,]_{\Gamma|\Delta}}$. We write $C[\,]_{\Gamma|\Delta}$ when the hole of C is annotated with $\Gamma \mid \Delta$. The translation $(-)^{\ddagger}$ of lambda-terms into hypernets in Figure 14 can then be extended accordingly, by translating the additional formation rule $\overline{\Gamma \mid \Delta \vdash [\,]_{\Gamma|\Delta}}$ into a path hypernet $\chi : \star \Rightarrow \star^{\otimes k} \otimes \diamond^{\otimes h}$, where k and h are the length of Γ and Δ respectively.

The translation of lambda-contexts yields hypernets that are binding-free contexts, and consequently, the observational equivalence $\simeq^{\ddagger}$ on lambda-terms is indeed a congruence relation with respect to lambda-contexts.

Lemma 11.3. *Let $\Gamma \mid \Delta \vdash t$ and $\Gamma \mid \Delta \vdash u$ be two derivable judgements. If $\Gamma \mid \Delta \vDash t \simeq^{\ddagger} u$ holds, then for any lambda-context C such that $\Gamma' \mid \Delta' \vdash_{\Lambda} C[\,]_{\Gamma|\Delta}$ is derivable, $\Gamma' \mid \Delta' \vDash C[t] \simeq^{\ddagger} C[u]$ holds.*

Proof outline. The proof is a combination of the congruence property of the contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}}_{\mathbb{N}\times\mathbb{N}}$ with respect to binding-free contexts, with two key properties of the translation of lambda-contexts. The two properties, stated below, can be proved by induction on lambda-contexts.

The first property is that the translation $(\overline{\Gamma' \mid \Delta' \vdash C[\,]_{\Gamma|\Delta}})^{\ddagger}$ is a binding-free context (as a hypernet). To check this property, one needs to examine paths to the sole source of the unique hole edge that appears in the translation. These paths are in fact always operation paths, noting that paths never go across the boundary of boxes (by Definition 6.2 (1)).

The second property is that $\Gamma' \mid \Delta' \vdash C[t]$ and $\Gamma' \mid \Delta' \vdash C[u]$ are both derivable, and moreover, their translations can be decomposed as follows:

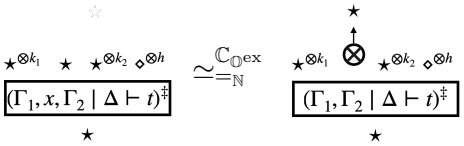
$$(\Gamma' \mid \Delta' \vdash C[t])^{\ddagger} = (\Gamma' \mid \Delta' \vdash C[\,]_{\Gamma \mid \Delta})^{\ddagger}[(\Gamma \mid \Delta \vdash t)^{\ddagger}],$$
$$(\Gamma' \mid \Delta' \vdash C[u])^{\ddagger} = (\Gamma' \mid \Delta' \vdash C[\,]_{\Gamma \mid \Delta})^{\ddagger}[(\Gamma \mid \Delta \vdash u)^{\ddagger}]. \qquad \square$$

12. Example equivalences

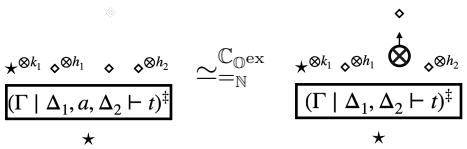
In this section we use the sufficiency-of-robustness theorem (Theorem 8.11) and prove some example equivalences. The first kind of example is *Weakening laws*.

Proposition 12.1 (Weakening laws)**.**

- *Given a derivable judgement* $\Gamma_1, x, \Gamma_2 \mid \Delta \vdash t$ *such that* $x \notin \mathrm{FV}(t)$*,*



- *Given a derivable judgement* $\Gamma \mid \Delta_1, a, \Delta_2 \vdash t$ *such that* $a \notin \mathrm{FA}(t)$*,*



where k, k_1, k_2, h, h_1, h_2 *are the lengths of* $\Gamma, \Gamma_1, \Gamma_2, \Delta, \Delta_1, \Delta_2$*, respectively.*

The next example equivalence is an instance of *parametricity*. This example originates in the Idealised Algol literature [OT97], a call-by-name language with ground-type *local* variables, although we state the example for the untyped call-by-value lambda-calculus extended primarily with store (Figure 13). Note that the example uses the standard call-by-value variable binding '`let`' and sequential composition ';', which are both defined by syntactic sugar (see Figure 13).

Proposition 12.2. *For any finite sequence of distinct variables* Γ *and any finite sequence of distinct variables* Δ*,*

$$\Gamma \mid \Delta \vDash \mathtt{let}\ x = \mathtt{ref}\ 1\ \mathtt{in}\ \lambda f.\,(f\ ();!x)\ \simeq^{\ddagger}\ \lambda f.\,(f\ ();1). \tag{12.1}$$

In the left hand side (lhs) of (12.1), a store is created by '`ref` 1', and any access to the store simply fetches the stored value, due to '$!x$', without modifying it. As a consequence, the fetched value is always expected to be the original stored value '1', and hence the whole computation involving the particular state is expected to have the same result as just having the value '1' in the first place as in the right hand side (rhs) of (12.1).

The equivalence (12.1) is a typical challenging example in the literature [OT97], which has been proved using *parametricity*. We take a different approach based on step-wise local reasoning. The equivalence is not an example that is meant to show the full power of our approach; it is a simple yet motivating example that requires building of a whole proof infrastructure and a non-trivial proof methodology.

The rest of this section is organised as follows. Section 12.1 shows prerequisites on the copying UAM $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$. Section 12.2 defines necessary pre-templates, and proves that these imply contextual refinements, using the sufficiency-of-robustness theorem (Theorem 8.11). Section 12.3 then combines the resultant contextual refinements to prove the example equivalences (Proposition 12.1 & Proposition 12.2). Finally, Section 12.4 describes design process of some of the pre-templates.

12.1. **Prerequisites.** Here we establish that the particular copying UAM $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$ is deterministic and refocusing, which enables us to apply the sufficiency-of-robustness theorem (Theorem 8.11).

Definition 12.3 (the behaviour $B_{\mathbb{O}^{\mathrm{ex}}}$)**.** The behaviour $B_{\mathbb{O}^{\mathrm{ex}}}$ is defined locally via the following rewrite rules.

- Rewrite rules for function application are in Figure 10b, where G_S is additionally required to be stable.
- Rewrite rules for reference manipulation are in Figure 19, where G_S is additionally required to be stable.
- Rewrite rules for arithmetic are in Figure 10a.

The extra requirement of stable hypernets reflects the call-by-value nature of the extended lambda-calculus.

Lemma 12.4. *The copying UAM $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$ is deterministic and refocusing.*

Proof. The proof boils down to show determinism, and preservation of the rooted property, of the compute transitions for $\mathbb{O}^{\mathrm{ex}}_{\lightning}$, thanks to Lemma 10.4.

Compute transitions of operations $\{\vec{@}, \mathtt{ref}, +, -, -_1\}$ are deterministic, because at most one rewrite rule can be applied to each state. In particular, the stable hypernet G_S in the figures is uniquely determined (by Lemma D.2(3)).

Compute transitions of name-accessing operations $\{=, :=, !\}$ are deterministic for the same reason as copy transitions (see the proof of Lemma 10.4).

Compute transitions of all the operations $\mathbb{O}^{\mathrm{ex}}_{\lightning}$ are stationary, and hence they preserve the rooted property. The stationary property can be checked using local rewrite rules. Namely, in each rewrite rule $\dot{H} \mapsto \dot{H}'$ of the operations, only one input of $|\dot{H}|$ has type $\star$, and $\dot{H} = \lightning; |\dot{H}|$ and $\dot{H}' = ?; |\dot{H}'|$. Moreover, any output of $|\dot{H}|$ with type $\star$ is a target of an atom edge or a box edge (by definition of stable hypernets), which implies $|\dot{H}|$ is one-way. □

Remark 12.5 (Requirement of stable hypernets in Definition 12.3)**.** Because any initial state is rooted, given that all transitions preserve the rooted property, we can safely assume that any state that arises in an execution is rooted. This means that the additional requirement of stable hypernets in Definition 12.3 is in fact guaranteed to be satisfied in any execution (by Lemma C.5, Lemma D.6 and Lemma D.4). ■

12.2. **Pre-templates and robustness.** We now define necessary pre-templates, which are the *parametricity* pre-template $\lhd^{\mathrm{Param}}$, two *operational* pre-templates $\lhd^{\mathtt{ref}}$ and $\lhd^{\vec{@}}$, and *structural* pre-templates.

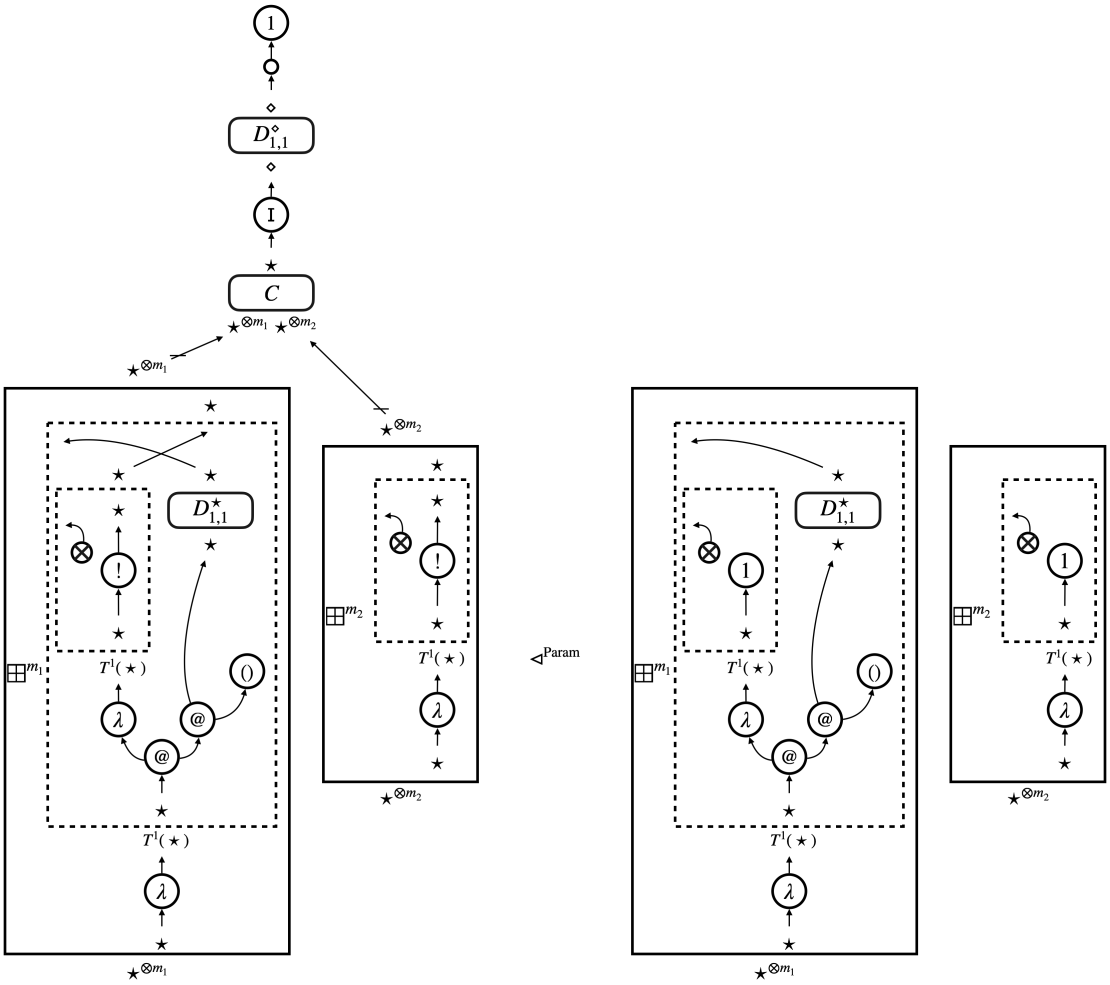


Figure 20: The parametricity pre-template where C is a contraction tree

Definition 12.6 (Pre-templates)**.**

- The *parametricity* pre-template $\lhd^{\mathrm{Param}}$ is as shown in Figure 20.
- The *micro-beta* pre-template $\lhd^{\vec{@}}$ is derived from rewrite rules as follows: $|\dot{G}_1| \lhd^{\vec{@}} |\dot{G}_2|$ if $\dot{G}_1 \mapsto \dot{G}_2$ is a micro-beta rewrite rule (Figure 10b) where G_S (see the figure) is not an arbitrary hypernet but a stable hypernet.
- The *reference-creation* pre-template $\lhd^{\mathtt{ref}}$ is derived from rewrite rules as follows: $|\dot{G}_1| \lhd^{\mathtt{ref}} |\dot{G}_2|$ if $\dot{G}_1 \mapsto \dot{G}_2$ is a reference-creation rewrite rule (Figure 19a) where G_S (see the figure) is not an arbitrary hypernet but a stable hypernet.
- *Structural* pre-templates are as shown in Figure 21, using the following notion of *box-permutation pair*.

Definition 12.7 (Box-permutation pair)**.** For any $n, k, h \in \mathbb{N}$, let ρ and ρ' be bijections on sets $\{1, \ldots, n{+}k{+}h\}$ and $\{1, \ldots, k{+}h\}$, respectively. These bijections form a *box-permutation* pair (ρ, ρ') if, for each $i \in \{1, \ldots, n + k + h\}$, the following holds:

(1) $\rho(i) = i$ if $1 \leq i \leq n$,

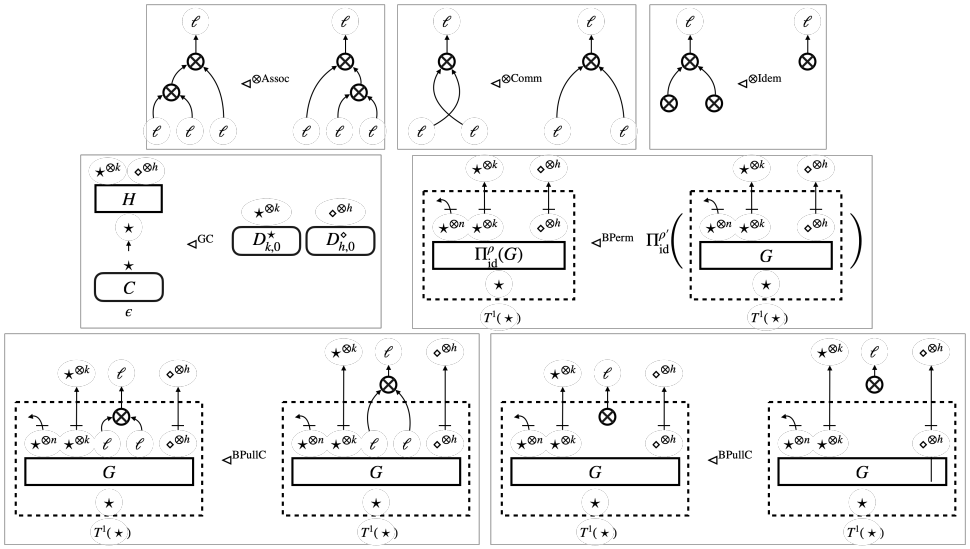


Figure 21: Structural pre-templates where $C : \epsilon \Rightarrow \star$ is a contraction tree, H is a copyable hypernet, G is a hypernet, and (ρ, ρ') is a box-permutation pair

(2) $\rho(i) = \rho'(i-n)$ if $n < i \leq n+k+h$,
(3) $1 \leq \rho'(i-n) \leq k$ if $n < i \leq n+k$,
(4) $k < \rho'(i-n) \leq k+h$ if $n+k < i \leq n+k+h$.

The rhs of $\lhd^{\mathrm{Param}}$ is straightforward. It simply consists of a bunch of encodings of two function abstractions, namely '$\lambda f.\,(\lambda w.\,1)\ (f\ ())$' and '$\lambda w.\,1$'. The empty store becomes absent in the graphical representation.

The lhs of $\lhd^{\mathrm{Param}}$ contains a bunch of encodings of two function abstractions, i.e. '$\lambda f.\,(\lambda w.\,!x)\ (f\ ())$' and '$\lambda w.\,!x$', and also the graphical representation of the store '$\{a \mapsto 1\}$' that consists of an atom edge and an edge labelled with the value '1'. The variable 'x' refers to the name 'a', and therefore, the encodings of function abstractions are all connected to the atom edge via a contraction tree.

All structural pre-templates (Figure 21) but $\lhd^{\mathrm{BPerm}}$ concern contractions and weakenings. The pre-template $\lhd^{\mathrm{BPerm}}$ lets us permute outputs of a box.

Thanks to Lemma 12.4, we can apply the sufficiency-of-robustness theorem (Theorem 8.11) and obtain contextual equivalences (on hypernets) as follows.

Lemma 12.8.

(1) *The micro-beta pre-template* $\lhd^{\vec{@}}$ *implies contextual equivalence* $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}}_{\mathbb{N}\times\mathbb{N}}$.
(2) *The reference-creation pre-template* $\lhd^{\mathtt{ref}}$ *implies contextual equivalence* $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}}_{\mathbb{N}\times\mathbb{N}}$.
(3) *Each structural pre-template implies contextual equivalence* $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{\mathbb{N}\times\mathbb{N}}$.

Proof outline. Table 3 summarises how we use Theorem 8.11 on the pre-templates. For example, $\lhd^{\otimes}$ is a $(\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}, \geq_{\mathbb{N}}, =_{\mathbb{N}})$-template, as shown in the "template" column, and both itself and its converse are $(\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}, =_{\mathbb{N}}, =_{\mathbb{N}}, =_{\mathbb{N}})$-robust relative to all rewrite transitions, as shown in the "robustness" columns. Thanks to monotonicity of robustness with respect to Q, we can then use Theorem 8.11(1) with a reasonable triple $(\geq_{\mathbb{N}}, =_{\mathbb{N}}, =_{\mathbb{N}})$, and Theorem 8.11(2)

	template (input-safety)	robustness of $\lhd$	robustness of $\lhd^{-1}$	dependency	implication of $H_1 \lhd H_2$
$\lhd^{\otimes\text{Assoc}}$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	—	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\otimes\text{Comm}}$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	—	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\otimes\text{Idem}}$	$\square, \square, \square$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	—	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\otimes}$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, \geq, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\lhd^{\otimes\text{Assoc}}$ $\lhd^{\otimes\text{Comm}}$	$H_1 \preceq_{\geq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$, $H_2 \preceq_{\leq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_1$
$\lhd^{\text{GC}}$	$\square, \square, \square$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	—	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\text{BPerm}}$	$\square, \square, \square$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	—	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\text{BPullC}}$	$\square, \square, \square$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\lhd^{\otimes\text{Assoc}}$ $\lhd^{\otimes\text{Comm}}$ $\lhd^{\otimes\text{Idem}}$	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\text{BPullW}}$	$\square, \square, \square$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =, =$	$\lhd^{\otimes\text{Idem}}$	$H_1 \simeq_{=\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
$\lhd^{\vec{@}}$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, \geq, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}, =, =, =$	—	$H_1 \preceq_{\geq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}} H_2$, $H_2 \preceq_{\leq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}} H_1$
	$\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}, \geq, =$				
$\lhd^{\texttt{ref}}$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, \geq, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}, =, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}, =, =, =$	—	$H_1 \preceq_{\geq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}} H_2$, $H_2 \preceq_{\leq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}} H_1$
	$\mathbb{C}_{\mathbb{0}^{\text{ex-bf}}}, \geq, =$				
$\lhd^{\text{Param}}$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, =, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, \geq, \geq, =$	$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, \leq, =, \leq$	$\lhd^{\otimes\text{Assoc}}$ $\lhd^{\otimes\text{Idem}}$ $\lhd^{\otimes}$ $\lhd^{\text{GC}}$	$H_1 \preceq_{\geq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$
		$\mathbb{C}_{\mathbb{0}^{\text{ex}}}, \geq, \geq, \geq$			$H_2 \preceq_{\leq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_1$

Table 3: Templates, with their robustness and implied contextual refinements/equivalences ($\square$ denotes anything)

with a reasonable triple $(\leq_{\mathbb{N}}, =_{\mathbb{N}}, =_{\mathbb{N}})$. Consequently, $H_1 \lhd^{\otimes} H_2$ implies $H_1 \preceq_{\geq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_2$ and $H_2 \preceq_{\leq\mathbb{N}}^{\mathbb{C}_{\mathbb{0}^{\text{ex}}}} H_1$, which is shown in the "implication of $H_1 \lhd H_2$" column.

Output-closure of the pre-templates can be easily checked, typically by spotting that an input or an output, of type $\star$, is a source or a target of a contraction, atom or box edge.

Pre-templates that relate hypernets with no input of type $\star$ are trivially a $(\mathbb{C}, Q, Q')$-template for any $\mathbb{C}$, Q and Q'. The table uses '$\square, \square, \square$' to represent this situation.

Typically, a reasonable triple for a pre-template can be found by selecting "bigger" parameters from those of input-safety and robustness, thanks to monotonicity of input-safety and robustness with respect to (Q, Q', Q''). However, the parametricity pre-template $\lhd^{\text{Param}}$ requires non-trivial use of the monotonicity. This is because the parameter $(\geq, \geq, =)$ that makes the pre-template robust, as the upper row in the "robustness" column shows, is not

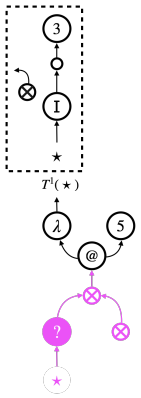
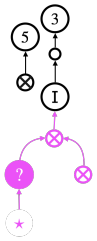
itself a reasonable triple. The lower row shows the alternative, bigger, parameter $(\geq, \geq, \geq)$ to which Theorem 8.11 can be applied.

In the table, cyan symbols indicate where a proof of input-safety or robustness relies on contextual refinement. The "dependency" column indicates which pre-templates can be used to prove the necessary contextual refinement, given that these pre-templates imply contextual refinement as shown elsewhere in the table. This reliance specifically happens in finding a quasi-specimen, using contextual refinements/equivalences via Lemma G.4. In the case of $\lhd^{\otimes}$, its input-safety and robustness are proved under the assumption that $\lhd^{\otimes\mathrm{Assoc}}$ and $\lhd^{\otimes\mathrm{Comm}}$ imply contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=_{\mathbb{N}}}$.

Detailed proofs of input-safety and robustness are in Section H.2 and Section H.3. □

Remark 12.9 (Necessity of binding-free contexts)**.** The restriction to binding-free contexts plays a crucial role only in robustness regarding the operational pre-templates $\lhd^{\vec{@}}$ and $\lhd^{\mathtt{ref}}$. In fact, these pre-templates are input-safe with respect to both $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$ and $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$. This gap reflects duplication behaviour on atom edges, which is only encountered in a proof of robustness.

In fact, robustness of the micro-beta pre-template $\lhd^{\vec{@}}$ is *not* guaranteed in the presence of copy transitions, which apply contraction rules. Starting from a pair of states given by a specimen $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$, it may be the case that some copy transitions are possible without reaching another (quasi-)specimen. An example scenario is when the specimen yields the following two states, where the context $\dot{\mathcal{C}}$ is indicated by magenta:

$\dot{\mathcal{C}}[\vec{H}^1] =$  , $\dot{\mathcal{C}}[\vec{H}^2] =$  .

Transitions from the state $\dot{\mathcal{C}}[\vec{H}^1]$ eventually duplicate the application edge (@), the abstraction edge (λ) and also the entire box connected to the abstraction edge. In particular, these transitions duplicate the atom edge contained in the box. However, transitions from the state $\dot{\mathcal{C}}[\vec{H}^2]$ can never duplicate the atom edge, because the edge is shallow in this state. This mismatch of duplication prevents the micro-beta pre-template from being robust relative to a copy transition.

This is why we restrict contexts to be binding-free, when it comes to robustness of operational pre-templates. If the context $|\dot{\mathcal{C}}|$ is binding-free, the situation explained above would never happen. Application of a contraction rule can involve the hypernets $\vec{H}^1$ and $\vec{H}^2$ only as a part of box contents that are duplicated as a whole. Any specimen of $\lhd^{\vec{@}}$ is therefore turned into another specimen whose context possibly has more holes, by a single copy transition. Note that this also explains why we allow the context of a specimen to have multiple holes. ■

	⊗ Assoc	⊗ Comm	⊗ Idem	⊗	GC	BPerm	BPullC	BPullW	$\vec{@}$	`ref`	Param
Proposition 12.1			○					○			
Proposition 12.2	○	○	○, (W)	●	●	○	○	○, (W)	○	○	○

Table 4: Dependency of example equivalences on templates

12.3. **Combining templates.** We now combine the contextual equivalences and prove Proposition 12.1 and Proposition 12.2. We start with combining the structural templates $\lhd^{\otimes\mathrm{Idem}}$ and $\lhd^{\mathrm{BPullW}}$, and prove the Weakening laws.

Proof outline of Proposition 12.1. The proof is by induction on derivations. Base cases are for variables, atoms and constants. The proof for these cases are trivial, because any distributor $D^{\ell}_{k,0}$ with no inputs is simply a bunch of weakening edges (see Definition 10.1).

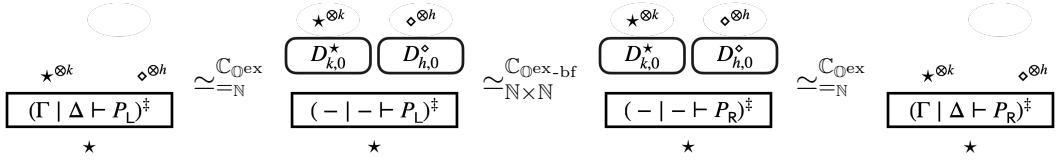
In inductive cases, we need to identify a single weakening edge with a certain (sub-)hypernet, namely: (i) a distributor $D^{\ell}_{1,1}$ whose sole input is connected to a weakening edge, (ii) a distributor $D^{\ell}_{1,2}$ whose two inputs are connected to weakening edges, and (iii) a distributor $D^{\ell}_{1,1}$ whose sole input is connected to a box edge, in which a weakening edge is connected to the corresponding output. The first two situations are for unary/binary operations and function application. These can be handled with the contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=\mathbb{N}}$ implied by $\lhd^{\otimes\mathrm{Idem}}$. The third situation is for function abstraction, and it boils down to the first situation, thanks to the contextual equivalence $\simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=\mathbb{N}}$ implied by $\lhd^{\mathrm{BPullW}}$. □

We can then prove the equivalence (12.1) as follows.

Proof of Proposition 12.2. Let P_{L} and P_{R} be the de-sugared version of the left-hand side and the right-hand side of (12.1), i.e.:

$$P_{\mathsf{L}} \equiv \big(\lambda x.\,\lambda f.\,(\lambda w.\,!x)\ (f\ ())\big)\ (\mathtt{ref}\ 1)$$
$$P_{\mathsf{R}} \equiv \lambda f.\,(\lambda w.\,1)\ (f\ ()).$$

The equivalence (12.1) can be obtained as a chain of contextual equivalences whose outline is as follows.



(12.2)

The leftmost and rightmost contextual equivalences are consequences of the Weakening laws (Proposition 12.1), because the terms P_{L} and P_{R} have no free variables nor free atoms. The middle contextual equivalence follows from the special case of (12.1) where the environment is empty, i.e. $- \mid - \vDash P_{\mathsf{L}} \simeq^{\ddagger} P_{\mathsf{R}}$. This contextual equivalence is namely derived from another chain of contextual equivalences that is shown in Figure 22, via the binding-free context that consists of a hole of type $\star \Rightarrow \epsilon$ and weakening edges (i.e. $D^{\star}_{k,0}$ and $D^{\diamond}_{h,0}$). In Figure 22, each contextual equivalence is accompanied by relevant templates, and preorders on natural numbers are omitted. □

This concludes our development leading to the proof our example equivalences.

Table 4 summarises dependency of the example equivalences on templates, which can be observed in the above proofs. The symbol '○' indicates *direct* dependency on templates, in

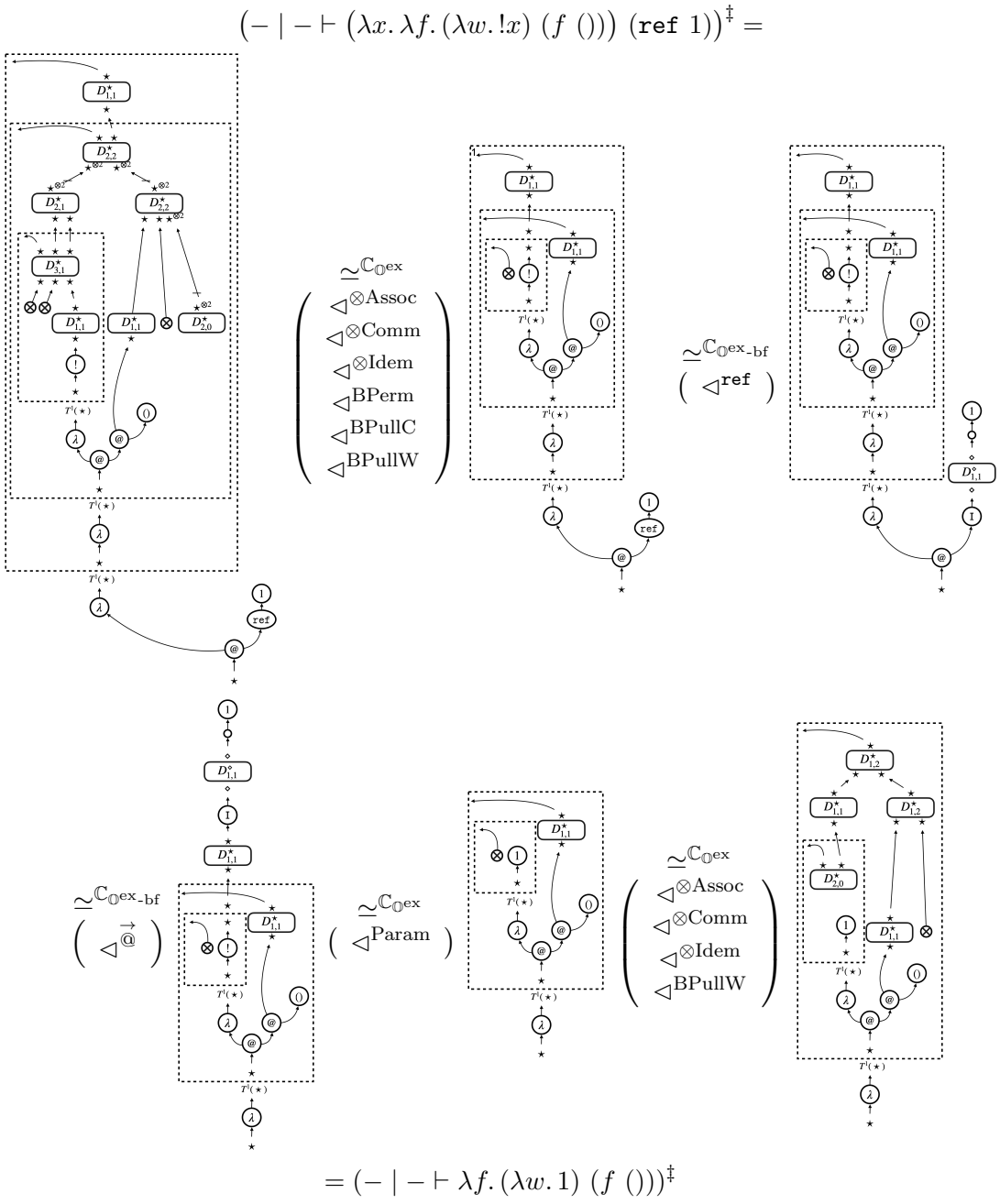


Figure 22: A proof illustration of Proposition 12.2, for the empty environment

the sense that an equivalence can be proved by combining contextual equivalences implied by these templates. For example, the Weakening laws are obtained by combining two templates $\lhd^{\otimes\mathrm{Idem}}$ and $\lhd^{\mathrm{BPullW}}$. Because the proof of Proposition 12.2 uses the Weakening law, it also depends on these two templates, which is indicated by '(W)' in the table.

Chosen terms and contexts:

$P_{\mathsf{L}} \equiv \big(\lambda x.\, \lambda f.\, (\lambda w.\, !x)\ (f\ ())\big)\ (\texttt{ref}\ 1)$ (de-sugared version of the lhs of the law)

$P_{\mathsf{R}} \equiv \lambda f.\, (\lambda w.\, 1)\ (f\ ())$ (de-sugared version of the rhs of the law)

$C_1 \equiv \big(\lambda g.\, (g\ (\lambda y.\, y)) + (g\ (\lambda z.\, 0))\big)\ [\,]$

Informal reduction of a program $C_1[P_{\mathsf{L}}]$ with empty store $\emptyset$:

L0	$\big(\lambda g.\, (g\ (\lambda y.\, y)) + (g\ (\lambda z.\, 0))\big)\ \Big(\big(\lambda x.\, \lambda f.\, (\lambda w.\, !x)\ (f\ ())\big)\ (\texttt{ref}\ 1)\Big)$	,	$\emptyset$	$\dashrightarrow$
L0'	$\big(\lambda g.\, (g\ (\lambda y.\, y)) + (g\ (\lambda z.\, 0))\big)\ \Big(\big(\lambda x.\, \lambda f.\, (\lambda w.\, !x)\ (f\ ())\big)\ a\Big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L1	$\big(\lambda g.\, (g\ (\lambda y.\, y)) + (g\ (\lambda z.\, 0))\big)\ \big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L2	$\big(\big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)\ (\lambda y.\, y)\big) + \big(\big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L3	$\big((\lambda w.\, !a)\ ((\lambda y.\, y)\ ())\big) + \big(\big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L4	$\big((\lambda w.\, !a)\ ()\big) + \big(\big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L4'	$!a + \big(\big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L5	$1 + \big(\big(\lambda f.\, (\lambda w.\, !a)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L6	$1 + \big((\lambda w.\, !a)\ ((\lambda z.\, 0)\ ())\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L7	$1 + \big((\lambda w.\, !a)\ 0\big)$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L7'	$1 + !a$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L8	$1 + 1$	,	$\{a \mapsto 1\}$	$\dashrightarrow$
L9	2	,	$\{a \mapsto 1\}$	

Informal reduction of a program $C_1[P_{\mathsf{R}}]$ with empty store $\emptyset$:

R1	$\big(\lambda g.\, (g\ (\lambda y.\, y)) + (g\ (\lambda z.\, 0))\big)\ \big(\lambda f.\, (\lambda w.\, 1)\ (f\ ())\big)$	,	$\emptyset$	$\dashrightarrow$
R2	$\big(\big(\lambda f.\, (\lambda w.\, 1)\ (f\ ())\big)\ (\lambda y.\, y)\big) + \big(\big(\lambda f.\, (\lambda w.\, 1)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$,	,	$\emptyset$	$\dashrightarrow$
R3	$\big((\lambda w.\, 1)\ ((\lambda y.\, y)\ ())\big) + \big(\big(\lambda f.\, (\lambda w.\, 1)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\emptyset$	$\dashrightarrow$
R4	$\big((\lambda w.\, 1)\ ()\big) + \big(\big(\lambda f.\, (\lambda w.\, 1)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\emptyset$	$\dashrightarrow$
R5	$1 + \big(\big(\lambda f.\, (\lambda w.\, 1)\ (f\ ())\big)\ (\lambda z.\, 0)\big)$	,	$\emptyset$	$\dashrightarrow$
R6	$1 + \big((\lambda w.\, 1)\ ((\lambda z.\, 0)\ ())\big)$	,	$\emptyset$	$\dashrightarrow$
R7	$1 + \big((\lambda w.\, 1)\ 0\big)$	,	$\emptyset$	$\dashrightarrow$
R8	$1 + 1$	,	$\emptyset$	$\dashrightarrow$
R9	2	,	$\emptyset$	

Figure 23: The equivalence (12.1): an example scenario

Note that two templates $\lhd^{\otimes}$ and $\lhd^{\mathrm{GC}}$ do not directly appear in the proof of Proposition 12.2, in particular in the chain shown in Figure 22. They are however necessary for robustness of the parametricity template (see Table 3). Proposition 12.2 depends on the two templates only *indirectly*, which is indicated by the symbol '•' in Table 4.

12.4. **Designing pre-templates.** We conclude this section with informal description of how some of the pre-templates are designed, using conventional reduction semantics. This description will reveal how graphical representation is better suited for local reasoning compared to textual representation (see e.g. (12.3)).

Instead of turning (12.1) directly into a single pre-template, we decompose the equivalence into several, more primitive, pre-templates. Once we apply the sufficiency-of-robustness

theorem to the pre-templates, the obtained contextual equivalences can be composed to yield (12.1), as we saw in Section 12.3. This approach increases the possibility of reusing parts of a proof of one law in a proof of another law. For example, most of the pre-templates that are used to prove (12.1) can be reused for proving the call-by-value Beta law (see [Mur20, Section 4.5] for details).

The idea of the pre-templates is that they can describe all the possible differences that may arise during execution of any two programs whose differences is given precisely by (12.1). As an illustration of the design process, we compare informal reduction sequences of two example programs, as summarised in Figure 23.

We choose the context C_1, which expects a function in the hole and uses it twice. It generates two programs, by receiving the terms P_L and P_R that are a de-sugared version of the two sides of (12.1).

Each informal reduction step $\dashrightarrow$ updates a term and its associated store. The step is either the standard call-by-value beta-reduction, addition of numbers (for '+'), reference creation (for '`ref`'), or dereferencing (for '!'). Reference creation is the only step that modifies store. It replaces a (sub-)term of the form '`ref` n' with a fresh name, say 'a', and extends the store with '$a \mapsto n$'. The empty store is denoted by '$\emptyset$'. Each reduction step in Figure 23 is given a tag, such as `L0`, `L0'`, `R1`. We use the tags for referring to a corresponding term and store, and also to the reduction step, if any, from the term and store.

Before explaining the colouring scheme of Figure 23, let us observe the (in-)correspondence between the reduction sequences of $C_1[P_\mathsf{L}]$ and $C_1[P_\mathsf{R}]$. The reduction sequence of $C_1[P_\mathsf{R}]$ consists of seven beta-reduction steps (`R1`–`R7`) and one addition step (`R8`). The other reduction sequence, of $C_1[P_\mathsf{L}]$, in fact contains steps that correspond to these seven beta-reduction steps and the addition step, as suggested by the tags (`L1`, `L2`, `L3`, `L4`, `L5`, `L6`, `L7`, `L8`). The sequence has four additional steps, namely: one reference-creation step (`L0`), one application step (`L0'`) of a function to the created name 'a', and two dereferencing steps (`L4'`, `L7'`). The two sequences result in the same term, but in different store (`L9`, `R9`).

The colouring scheme is as follows. In the eight matching steps and the final result, differences between two sides (e.g. `L1` and `R1`) are highlighted in magenta. Note that it is not the minimum difference that are highlighted. Highlighted parts are chosen in such a way that they capture the smallest difference "on the surface". Sub-terms *on the surface* are those that are outside of any lambda-abstraction, which can be graphically represented by sub-hypernets that are outside of any boxes. For example, after the name creation (i.e. in `L1`, `R1`), the function abstractions ('$\lambda f.\,(\lambda w.\,!a)\ (f\ ())$', '$\lambda f.\,(\lambda w.\,1)\ (f\ ())$') are highlighted, instead of the minimum difference ('$!a$', '1').

In summary, the reduction sequences of $C_1[P_\mathsf{L}]$ and $C_1[P_\mathsf{R}]$ have eight steps corresponding with each other (`L1`–`L8`, `R1`–`R8`) where the differences of their results can be described by means of store and sub-terms on the surface. The extracted differences are namely function abstractions ('$\lambda f.\,(\lambda w.\,!a)\ (f\ ())$' and '$\lambda w.\,!a$', against '$\lambda f.\,(\lambda w.\,1)\ (f\ ())$' and '$\lambda w.\,1$') and store ('$\{a \mapsto 1\}$' against '$\emptyset$'). By simply collecting these differences, we can obtain our first pre-template: *parametricity* pre-template $\lhd^{\mathrm{Param}}$. It is the essential, key, pre-template for (12.1). Textually, and intuitively, it looks like the following.

$$\begin{aligned}&\lambda f.\,(\lambda w.\,!a)\ (f\ ()),\ \ldots,\ \lambda f.\,(\lambda w.\,!a)\ (f\ ()),\ \lambda w.\,!a,\ \ldots,\ \lambda w.\,!a,\ \{a \mapsto 1\}\\ \lhd^{\mathrm{Param}}\quad &\lambda f.\,(\lambda w.\,1)\ (f\ ()),\ \ldots,\ \lambda f.\,(\lambda w.\,1)\ (f\ ()),\ \lambda w.\,1,\ \ldots,\ \lambda w.\,1,\ \emptyset\end{aligned} \tag{12.3}$$

Graphically, the parametricity pre-template is simply a relation between hypernets (Figure 20). In particular, the lhs of the pre-template can be represented as a single hypernet, thanks to the graphical representation where mentions of a name 'a' become connection. Hypernets of the function abstractions are all connected to the hypernet of the store '$\{a \mapsto 1\}$'. This graphical representation naturally entails the crucial piece of information, which is invisible in the informal textual representation (12.3), that the function abstractions are the all and only parts of a program that have access to the name 'a'.

The choice of smallest differences on the surface, instead of absolute minimum difference, also plays a key role here. Should we choose minimum differences that may be inside lambda-abstraction, their hypernet representations cannot be directly combined to yield a single valid hypernet. This is due to the box structure of hypernets, which are used to represent function abstraction. In other words, to connect the hypernet of the store '$\{a \mapsto 1\}$' to the hypernet of the sub-term '$!a$' that appears inside the function abstraction '$\lambda w.\, !a$', we must first make a connection to the hypernet of the whole function abstraction, which contains the hypernet of '$!a$' inside a box.

Recall that the parametricity pre-template $\lhd^{\mathrm{Param}}$ collects differences in matching steps (eight steps each, i.e. `L1`–`L8`, `R1`–`R8`). From the first two unmatched steps (`L0`, `L0'`), we extract the two operational pre-templates $\lhd^{\mathtt{ref}}$ and $\lhd^{\vec{@}}$. These are both induced by the reductions, namely: $\lhd^{\mathtt{ref}}$ by reference creation (`L0`) and $\lhd^{\vec{@}}$ by beta-reduction (`L0'`). For instance, these pre-templates relate the sub-terms that are yielded by the two reductions (`L0`, `L0'`), informally as follows.

$$\begin{array}{rcl} \mathtt{ref}\ 1,\ \emptyset & \lhd^{\mathtt{ref}} & a,\ \{a \mapsto 1\} \\ \big(\lambda x.\, \lambda f.\, (\lambda w.\, !x)\ (f\ ())\big)\ a & \lhd^{\vec{@}} & \lambda f.\, (\lambda w.\, !a)\ (f\ ()) \end{array}$$

The three pre-templates $\lhd^{\mathrm{Param}}$, $\lhd^{\mathtt{ref}}$, $\lhd^{\vec{@}}$ are all the key pre-templates we needed for (12.1). Once the sufficiency-of-robustness theorem is applied to these pre-templates, the equivalence can be obtained as a chain of the induced contextual equivalences. The chain roughly looks as follows for the particular programs $C_1[P_{\mathsf{L}}]$ and $C_1[P_{\mathsf{R}}]$, where we use $\simeq$ to denote the informal textual counterpart of contextual equivalence that is between terms accompanied by store.

$$\begin{array}{rcll} C_1[P_{\mathsf{L}}],\ \emptyset & \equiv & C_1[\big(\lambda x.\, \lambda f.\, (\lambda w.\, !x)\ (f\ ())\big)\ (\mathtt{ref}\ 1)],\ \emptyset & \\ & \simeq & C_1[\big(\lambda x.\, \lambda f.\, (\lambda w.\, !x)\ (f\ ())\big)\ a],\ \{a \mapsto 1\} & (\text{induced by } \lhd^{\mathtt{ref}}) \\ & \simeq & C_1[\lambda f.\, (\lambda w.\, !a)\ (f\ ())],\ \{a \mapsto 1\} & (\text{induced by } \lhd^{\vec{@}}) \\ & \simeq & C_1[\lambda f.\, (\lambda w.\, 1)\ (f\ ())],\ \emptyset & (\text{induced by } \lhd^{\mathrm{Param}}) \\ & \equiv & C_1[P_{\mathsf{R}}],\ \emptyset & \end{array} \tag{12.4}$$

Finally, in the formal proof that uses hypernets and focussed graph rewriting rather than terms and reductions, we additionally needed auxiliary, structural pre-templates (Figure 21). These enable us to simplify or identify certain contractions and weakenings, which are not present in the textual representation but important in the graphical representation as hypernets. Contextual equivalences induced by the structural pre-templates enable simplification of certain hypernets that involve contractions and weakenings. The simplification appeared in the formal counterpart of the chain (12.4), which is shown in Figure 22. It primarily

applies to the hypernets produced by the encoding $(-)^{\ddagger}$. Additionally, the pre-templates helped us prove input-safety and robustness of other pre-templates.

13. Related and future work

13.1. **Proof methodologies for observational equivalence.** This work deals with fragility of observational equivalence and of its proof methodologies. Dreyer et al. address this issue by carefully distinguishing between various kinds of operations (state vs. control) [DNB12]. Dreyer et al. [DNRB10] use Kripke relations to go beyond an enumerative classification of effects; they use *characterisation* of effects in the aid of reasoning. Their notion of *island* has similar intuitions to our robustness property. More radical approaches are down to replacing the concept of syntactic context with an *epistemic* context, akin to a Dolev-Yao-style attacker [GT12], and characterising combinatorially the interaction between a term and its context as is the case with the game semantics [AJM00, HO00] or the trace semantics [JR05]. We propose a new approach to the problem of fragility, namely directly reasoning about *robustness* of observational equivalence, using a uniform graph representation of a program and its context.

We are not the first one to take a coinductive approach to observational equivalence. *Applicative bisimilarity* and its successor, *environmental bisimilarity*, have been successfully used to prove observational equivalence in various effectful settings [Abr90, KLS11, DGL17, SV18]. Typically, one first constructs an applicative (or environmental) bisimulation, and then proves it is a congruence using Howe's method [How96]. In contrast, in our approach, one first constructs a relation that is closed, by definition, under term (graph) construction, and then proves it is a counting simulation up-to. Our approach does not need Howe's method.

We argue that our approach is both flexible and elementary. A specific version [MCG18] of this formalism has been used to prove, for example, the soundness of exotic operations involved in (a functional version of) Google's TensorFlow machine learning library. Even though the proofs can seem complicated, this is in part due to the graph-based formalism being new, and in part due to the fact that proofs of equivalences are lengthy case-based analyses. However, herein lies the simplicity and the robustness of the approach, avoiding the discovery of clever-but-fragile language invariants which can be used to streamline proofs.

Our tedious-but-elementary proofs on the other hand seem highly suitable for automation. The idea of elementary case analysis can be adopted in term rewriting, instead of hypernet rewriting, for a call-by-value lambda-calculus equipped with *effect handlers* [Pre15]. In this setting, the case analysis can be formalised as *critical pair analysis*, which is a fundamental and automatable technique in term rewriting, and indeed automated [MH24].

13.2. **Focussed hypernet rewriting.** Focussed hypernet rewriting is a radically new approach to defining effectful programming languages and proving observational equivalence. We are not so much interested in *simulated* effects, which are essentially the encoding of effectful behaviour into pure languages, and which can be achieved via monads [Wad98], but we are interested in genuine *native* effects which happen outside of the language. Semantically this has been introduced by Plotkin and Power [PP08] and more recently developed by Møgelberg and Staton [MS11]. The (copying) UAM takes the idea to the extreme by situating all operations (pure or effectful) outside of the primitives, and by keeping as

intrinsic to the language only the structural aspects of copying vs. sharing, and scheduling of computation via thunking.

The (copying) UAM presented in this paper is an extension of the *Dynamic Geometry of Interaction Machine (DGoIM)* [MG17] to effects. The DGoIM builds on operational machinery [DR96, Mac95, HMH14, MHH16] inspired by Girard's *Geometry of Interaction (GoI)* [Gir89]. Unlike these "conventional" GoI-inspired operational semantics, the DGoIM and hence the UAM are modified so that the underlying hypernet can be rewritten during execution.

The original motivation of the DGoIM was to produce an abstract machine that expresses the computational intuitions of the GoI while correctly modelling the cost of evaluation, particularly for call-by-value and call-by-need. The ability to rewrite its own hypernet makes the DGoIM *efficient*, in the sense of Accattoli et al. [ABM14], for common reduction strategies (namely, call-by-value and call-by-need). It also gives the DGoIM the ability to model exotic effects, e.g. transforming stateful into pure computation by abstracting the state [MCG18]. As an extension of the DGoIM, the UAM is designed for new reasoning principles and methods that arise out of GoI-inspired operational semantics.

Although the UAM does not aim at efficiency at the moment, one can think of a cost model of the UAM in a similar way as the DGoIM. Moreover, the indexing of observational equivalence with a preorder representing the number of steps gives a direct avenue for modelling and comparing computation costs. For example, the micro-beta law induced by the pre-template $\lhd^{\vec{@}}$ is indexed by the normal order $\geq$ on $\mathbb{N}$ (cf. Table 3), which indicates that one side always requires fewer steps than the other in the evaluation process. The only details to be resolved are associating costs (time and space) with steps, in particular different costs for different operations.

The UAM is motivated by a need for a flexible and expressive framework in which a wide variety of effects can be given a cost-accurate model. As discussed, the UAM opens the door to a uniform study of operations and their interactions. Defining new styles of abstract machines is a rich and attractive vein of research. The *monoidal computer* of Pavlovic [Pav13] or the *evolving algebras* of Gurevich [Gur18] are such examples. What sets the UAM apart is the fact that it can be used, rather conveniently, for reasoning robustly about observational equivalence.

13.3. **Hypernets.** The hierarchy of hypernets is inspired by the *exponential boxes* of proof nets, a graphical representation of linear logic proofs [Gir87] and have an informal connection to Milner's *bigraphs* [Mil01]. Exponential boxes can be formalised by parameterising an agent (which corresponds to an edge in our setting) by a net, as indicated by Lafont [Laf95]. In the framework of interaction nets [Laf90] that subsume proof nets, agents can be coordinated to represent a boundary of a box, as suggested by Mackie [Mac98]. An alternative representation of boxes that use extra links between agents is proposed by Accattoli and Guerrini [AG09].

Our graphical formulation of boxes shares the idea with the first parameterising approach, but we have flexibility regarding types of a box edge itself and its content (i.e. the hypernet that labels it). We use box edges to represent thunks, and a box edge can have less targets than outputs of its contents, reflecting the number of bound variables a thunk has. This generalised box structure is also studied by Drewes et al. [DHP02] as *hierarchical graphs*, in the context of double-pushout graph transformation (DPO) [Roz97], an well-established algebraic approach to graph rewriting. More recently, Alvarez-Picallo et al. have formulated

DPO rewriting for a class of hypernets similar to those used here [AGSZ22]; their work further relates hypernets with string diagrams with *functorial boxes* in the style of Melliès [Mel06].

Interaction nets are another established framework of graph rewriting, in which various evaluations of pure lambda-calculus can be implemented [Sin05, Sin06]. The idea of having the token to represent an evaluation strategy can be found in *loc. cit.*, which suggests that our focussed rewriting on hypernets could be implemented using interaction nets. However, the local reasoning we are aiming at with focussed rewriting does not seem easy in the setting of interaction nets, because of technical subtleties observed in *loc. cit.*; namely, a status of evaluation is remembered by not only the token but also some other agents around an interaction net.

13.4. **Future work.** One direction is the introduction of a more meaningful type system for hypernets. The current type system of hypernets is very weak, just ensuring well-formedness. We consider it a strength of the approach that equivalences can be proved without the aid of a powerful type infrastructure. On the other hand, in order to avoid stuck configurations and ensure safety of evaluation, more expressive types are required. The usage of more expressive types is perfectly compatible with focussed hypernet rewriting, and is something we intend to explore. In particular we would like to study notions of typing which are germane to focussed hypernet rewriting, capturing its concepts of locality and robustness.

Beyond types, if we look at logics there are some appealing similarities between hypernet rewriting and *separation logic* [Rey02]. The division of nodes into copying nodes via variables and sharing nodes via atoms is not accidental, and their different contraction properties match those from *bunched implications* [OP99]. On a deeper level, the concepts of locality and in particular robustness developed here are related to the *frame* rule of separation logic.

Finally, our formulation of equivalence has some self-imposed limitations needed to limit the complexity of the technical presentation. We are hereby concerned with *sequential* and *deterministic* computation. Future work will show how these restrictions can be relaxed. Parallelism and concurrency can be naturally simulated using multi-token reductions, as inspired by the multi-token GoI machine of Dal Lago et al. [DTY17], whereas nondeterminism (or probabilistic evaluation) requires no changes to the machinery but rather a new definition of observational equivalence. This is work we are aiming to undertake in the future. A first step towards nondeterminism has been made [MSU24] in which the notion of counting simulation is extended from branching-free transition systems to nondeterministic automata.

References

[ABM14] Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In *19th ACM SIGPLAN International Conference on Functional Programming, ICFP 2014, September 1-3 2014, Gothenburg, Sweden*, pages 363–376, 2014. doi:10.1145/2628136.2628154.

[Abr90] Samson Abramsky. *The lazy lambda-calculus*, page 65–117. Addison Wesley, 1990.

[Abr97] Samson Abramsky. Game semantics for programming languages. In *22nd International Symposium on Mathematical Foundations of Computer Science, MFCS 1997, August 25-29 1997, Bratislava, Slovakia*, pages 3–4, 1997. doi:10.1007/BFb0029944.

[ADV20] Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The machinery of interaction. In *PPDP 2020*, pages 4:1–4:15. ACM, 2020.

[ADV21] Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The (in)efficiency of interaction. *Proc. ACM Program. Lang.*, 5(POPL):1–33, 2021. doi:10.1145/3434332.

[AG09] Beniamino Accattoli and Stefano Guerrini. Jumping boxes. In *CSL 2009*, volume 5771 of *Lect. Notes Comp. Sci.*, pages 55–70. Springer, 2009. doi:10.1007/978-3-642-04027-6_7.

[AGSZ22] Mario Alvarez-Picallo, Dan R. Ghica, David Sprunger, and Fabio Zanasi. Rewriting for monoidal closed categories. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 29:1–29:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.FSCD.2022.29.

[AGSZ23] Mario Alvarez-Picallo, Dan R. Ghica, David Sprunger, and Fabio Zanasi. Functorial string diagrams for reverse-mode automatic differentiation. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic, CSL 2023, February 13-16, 2023, Warsaw, Poland*, volume 252 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CSL.2023.6.

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000. doi:10.1006/INCO.2000.2930.

[BGK+22a] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. String diagram rewrite theory I: rewriting with frobenius structure. *J. ACM*, 69(2):14:1–14:58, 2022. doi:10.1145/3502719.

[BGK+22b] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. String diagram rewrite theory II: rewriting with symmetric monoidal structure. *Math. Struct. Comput. Sci.*, 32(4):511–541, 2022. doi:10.1017/S0960129522000317.

[BGK+22c] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski, and Fabio Zanasi. String diagram rewrite theory III: confluence with and without frobenius. *Math. Struct. Comput. Sci.*, 32(7):829–869, 2022. doi:10.1017/S0960129522000123.

[BPPR17] Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. A general account of coinduction up-to. *Acta Inf.*, 54(2):127–190, 2017. doi:10.1007/s00236-016-0271-4.

[DGL17] Ugo Dal Lago, Francesco Gavazzo, and Paul Blain Levy. Effectful applicative bisimilarity: Monads, relators, and howe's method. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005117.

[DHP02] Frank Drewes, Berthold Hoffmann, and Detlef Plump. Hierarchical graph transformation. *J. Comput. Syst. Sci.*, 64(2):249–283, 2002. doi:10.1006/jcss.2001.1790.

[DMMZ12] Olivier Danvy, Kevin Millikin, Johan Munk, and Ian Zerny. On inter-deriving small-step and big-step semantics: A case study for storeless call-by-need evaluation. *Theor. Comput. Sci.*, 435:21–42, 2012. doi:10.1016/j.tcs.2012.02.023.

[DNB12] Derek Dreyer, Georg Neis, and Lars Birkedal. The impact of higher-order state and control effects on local relational reasoning. *J. Funct. Program.*, 22(4-5):477–528, 2012. doi:10.1017/S095679681200024X.

[DNRB10] Derek Dreyer, Georg Neis, Andreas Rossberg, and Lars Birkedal. A relational modal logic for higher-order stateful adts. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '10, pages 185–198, New York, NY, USA, 2010. ACM. doi:10.1145/1706299.1706323.

[DR96] Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal lambda-machines. *Elect. Notes in Theor. Comp. Sci.*, 3:40–60, 1996.

[DTY17] Ugo Dal Lago, Ryo Tanaka, and Akira Yoshimizu. The geometry of concurrent interaction: Handling multiple ports by way of multiple tokens. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005112.

[Ghi23] Dan R. Ghica. The far side of the cube. In Alessandra Palmigiano and Mehrnoosh Sadrzadeh, editors, *Samson Abramsky on Logic and Structure in Computer Science and Beyond*, pages 219–250. Springer Verlag, 2023.

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

[Gir89] Jean-Yves Girard. Geometry of Interaction I: interpretation of system F. In *Logic Colloquium 1988*, volume 127 of *Studies in Logic & Found. Math.*, pages 221–260. Elsevier, 1989.

[GT12] Dan R. Ghica and Nikos Tzevelekos. A system-level game semantics. *Electr. Notes Theor. Comput. Sci.*, 286:191–211, 2012. doi:10.1016/j.entcs.2012.08.013.

[Gur18] Yuri Gurevich. Evolving algebras 1993: Lipari guide. *arXiv preprint arXiv:1808.06255*, 2018.

[GZ23] Dan Ghica and Fabio Zanasi. String diagrams for λ-calculi and functional computation. *arXiv preprint arXiv:2305.18945*, 2023.

[HMH14] Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In *Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014, July 14-18 2014, Vienna, Austria*, pages 52:1–52:10, 2014. doi:10.1145/2603088.2603124.

[HO00] J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000. doi:10.1006/inco.2000.2917.

[How96] Douglas J. Howe. Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, 124(2):103–112, 1996. doi:10.1006/INCO.1996.0008.

[JR05] Alan Jeffrey and Julian Rathke. Java jr: Fully abstract trace semantics for a core java language. In *14th European Symposium on Programming, ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, April 4-8 2005, Edinburgh, UK*, pages 423–438, 2005. doi:10.1007/978-3-540-31987-0_29.

[KLS11] Vasileios Koutavas, Paul Blain Levy, and Eijiro Sumii. From applicative to environmental bisimulation. In Michael W. Mislove and Joël Ouaknine, editors, *Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics, MFPS 2011, Pittsburgh, PA, USA, May 25-28, 2011*, volume 276 of *Electronic Notes in Theoretical Computer Science*, pages 215–235. Elsevier, 2011. doi:10.1016/J.ENTCS.2011.09.023.

[Laf90] Yves Lafont. Interaction nets. In *17th Annual ACM Symposium on Principles of Programming Languages, POPL 1990, January 1990, San Francisco, California, USA*, pages 95–108, 1990. doi:10.1145/96709.96718.

[Laf95] Yves Lafont. *From proof nets to interaction nets*, page 225–248. London Mathematical Society Lecture Note Series. Cambridge University Press, 1995. doi:10.1017/CBO9780511629150.012.

[Mac95] Ian Mackie. The Geometry of Interaction machine. In *POPL 1995*, pages 198–208. ACM, 1995.

[Mac98] Ian Mackie. Linear logic *With* boxes. In *13th IEEE Symposium on Logic in Computer Science, June 21-24 1998, Indianapolis, IN, USA*, pages 309–320, 1998. doi:10.1109/LICS.1998.705667.

[MCG18] Koko Muroya, Steven W. T. Cheung, and Dan R. Ghica. The geometry of computation-graph abstraction. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12 2018*, pages 749–758, 2018. doi:10.1145/3209108.3209127.

[Mel06] Paul-André Melliès. Functorial boxes in string diagrams. In *20th International Workshop on Computer Science Logic, CSL 2006, 15th Annual Conference of the EACSL, September 25-29 2006, Szeged, Hungary*, pages 1–30, 2006. doi:10.1007/11874683_1.

[MG17] Koko Muroya and Dan R. Ghica. The dynamic geometry of interaction machine: A call-by-need graph rewriter. In *26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24 2017, Stockholm, Sweden*, pages 32:1–32:15, 2017. doi:10.4230/LIPIcs.CSL.2017.32.

[MH24] Koko Muroya and Makoto Hamana. Term evaluation systems with refinements: First-order, second-order, and contextual improvement. In Jeremy Gibbons and Dale Miller, editors, *Functional and Logic Programming - 17th International Symposium, FLOPS 2024, Kumamoto, Japan, May 15-17, 2024, Proceedings*, volume 14659 of *Lecture Notes in Computer Science*, pages 31–61. Springer, 2024. doi:10.1007/978-981-97-2300-3_3.

[MHH16] Koko Muroya, Naohiko Hoshino, and Ichiro Hasuo. Memoryful geometry of interaction II: recursion and adequacy. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20-22 2016*, pages 748–760, 2016. doi:10.1145/2837614.2837672.

[Mil01] Robin Milner. Bigraphical reactive systems. In *International Conference on Concurrency Theory*, pages 16–35. Springer, 2001. doi:10.1007/3-540-44685-0_2.

[MJ69] James Hiram Morris Jr. *Lambda-calculus models of programming languages.* PhD thesis, Massachusetts Institute of Technology, 1969. URL: https://dspace.mit.edu/handle/1721.1/64850.

[MS11] Rasmus Ejlers Møgelberg and Sam Staton. Linearly-used state in models of call-by-value. In *4th International Conference on Algebra and Coalgebra in Computer Science, CALCO*

2011, August 30 - September 2 2011, Winchester, UK, pages 298–313, 2011. doi:10.1007/978-3-642-22944-2_21.

[MSU24] Koko Muroya, Takahiro Sanada, and Natsuki Urabe. Preorder-constrained simulations for program refinement with effects. In *CMCS 2024*, 2024. To appear.

[Mur20] Koko Muroya. *Hypernet Semantics of Programming Languages*. PhD thesis, University of Birmingham, 2020.

[OP99] Peter W O'Hearn and David J Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999. doi:10.2307/421090.

[OT97] Peter O'Hearn and Robert Tennent, editors. *Algol-like languages*. Progress in theoretical computer science. Birkhauser, 1997. doi:10.1007/978-1-4612-4118-8.

[Pav13] Dusko Pavlovic. Monoidal computer I: Basic computability by string diagrams. *Information and Computation*, 226:94–116, 2013. doi:10.1016/j.ic.2013.03.007.

[Pit00] Andrew M. Pitts. Operational semantics and program equivalence. In Gilles Barthe, Peter Dybjer, Luís Pinto, and João Saraiva, editors, *Applied Semantics, International Summer School, APPSEM 2000, Caminha, Portugal, September 9-15, 2000, Advanced Lectures*, volume 2395 of *Lecture Notes in Computer Science*, pages 378–412. Springer, 2000. doi:10.1007/3-540-45699-6_8.

[PP08] Gordon D. Plotkin and John Power. Tensors of comodels and models for operational semantics. *Electr. Notes Theor. Comput. Sci.*, 218:295–311, 2008. doi:10.1016/j.entcs.2008.10.018.

[Pre15] Matija Pretnar. An introduction to algebraic effects and handlers. invited tutorial paper. *Elect. Notes in Theor. Comp. Sci.*, 319:19–35, 2015.

[Rey02] John C Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74. IEEE, 2002. doi:10.1109/LICS.2002.1029817.

[Roz97] Grzegorz Rozenberg, editor. *Handbook of graph grammars and computing by graph transformations, volume 1: foundations*. World Scientific, 1997.

[San95] David Sands. Total correctness by local improvement in program transformation. In Ron K. Cytron and Peter Lee, editors, *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, pages 221–232. ACM Press, 1995. doi:10.1145/199448.199485.

[Sin05] François-Régis Sinot. Call-by-name and call-by-value as token-passing interaction nets. In *7th International Conference on Typed Lambda Calculi and Applications, TLCA 2005, April 21-23 2005, Nara, Japan*, pages 386–400, 2005. doi:10.1007/11417170_28.

[Sin06] François-Régis Sinot. Call-by-need in token-passing nets. *Mathematical Structures in Computer Science*, 16(4):639–666, 2006. doi:10.1017/S0960129506005408.

[SKS07] Davide Sangiorgi, Naoki Kobayashi, and Eijiro Sumii. Environmental bisimulations for higher-order languages. In *LICS 2007*, pages 293–302. IEEE Computer Society, 2007.

[Sta85] Richard Statman. Logical relations and the typed lambda-calculus. *Information and Control*, 65(2/3):85–97, 1985. doi:10.1016/S0019-9958(85)80001-2.

[SV18] Alex Simpson and Niels F. W. Voorneveld. Behavioural equivalence via modalities for algebraic effects. In Amal Ahmed, editor, *Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10801 of *Lecture Notes in Computer Science*, pages 300–326. Springer, 2018. doi:10.1007/978-3-319-89884-1_11.

[Wad98] Philip Wadler. The marriage of effects and monads. In *ACM SIGPLAN Notices*, volume 34:1, pages 63–74. ACM, 1998. doi:10.1145/289423.289429.

Appendix A. An alternative definition of hypernets

Informally, hypernets are nested hypergraphs, and one hypernet can contain nested hypergraphs up to different depths. This intuition is reflected by Definition 3.6 of hypernets, in particular the big union in $\mathcal{H}_{k+1}(L,M) = \mathcal{H}\Big(L, M \cup \bigcup_{i\leq k} \mathcal{H}_i(L,M)\Big)$. In fact, the definition can be replaced by a simpler, but possibly less intuitive, definition below that does not explicitly deal with the different depths of nesting.

Definition A.1. Given sets L and M, a set $\mathcal{H}'_k(L,M)$ is defined by induction on $k \in \mathbb{N}$:

$$\begin{aligned}\mathcal{H}'_0(L,M) &:= \mathcal{H}(L,M)\\ \mathcal{H}'_{k+1}(L,M) &:= \mathcal{H}\Big(L, M \cup \mathcal{H}'_k(L,M)\Big)\end{aligned}$$

and hence a set $\mathcal{H}'_\omega(L,M) := \bigcup_{i\in\mathbb{N}} \mathcal{H}'_i(L,M)$.

Lemma A.2. *Given arbitrary sets L and M, any two numbers $k, k' \in \mathbb{N}$ satisfy $\mathcal{H}'_k(L,M) \subseteq \mathcal{H}'_{k+k'}(L,M)$.*

Proof. If $k' = 0$, the inclusion trivially holds. If not, i.e. $k' > 0$, it can be proved by induction on $k \in \mathbb{N}$. The key reasoning principle we use is that $M \subseteq M'$ implies $\mathcal{H}(L,M) \subseteq \mathcal{H}(L,M')$.

In the base case, when $k = 0$ (and $k' > 0$), we have

$$\begin{aligned}\mathcal{H}'_0(L,M) &= \mathcal{H}(L,M)\\ &\subseteq \mathcal{H}\Big(L, M \cup \mathcal{H}'_{k'-1}(L,M)\Big) = \mathcal{H}'_{k'}(L,M).\end{aligned}$$

In the inductive case, when $k > 0$ (and $k' > 0$), we have

$$\begin{aligned}\mathcal{H}'_k(L,M) &= \mathcal{H}\Big(L, M \cup \mathcal{H}'_{k-1}(L,M)\Big)\\ &\subseteq \mathcal{H}\Big(L, M \cup \mathcal{H}'_{k-1+k'}(L,M)\Big) = \mathcal{H}'_{k+k'}(L,M)\end{aligned}$$

where the inclusion is by induction hypothesis on $k-1$. □

Proposition A.3. *Any sets L and M satisfy $\mathcal{H}_k(L,M) = \mathcal{H}'_k(L,M)$ for any $k \in \mathbb{N}$, and hence $\mathcal{H}_\omega(L,M) = \mathcal{H}'_\omega(L,M)$.*

Proof. We first prove $\mathcal{H}_k(L,M) \subseteq \mathcal{H}'_k(L,M)$ by induction on $k \in \mathbb{N}$. The base case, when $k = 0$, is trivial. In the inductive case, when $k > 0$, we have

$$\begin{aligned}\mathcal{H}_k(L,M) &= \mathcal{H}\Big(L, M \cup \bigcup_{i\leq k-1} \mathcal{H}_i(L,M)\Big)\\ &\subseteq \mathcal{H}\Big(L, M \cup \bigcup_{i\leq k-1} \mathcal{H}'_i(L,M)\Big) && \text{(by I.H.)}\\ &= \mathcal{H}\Big(L, M \cup \mathcal{H}'_{k-1}(L,M)\Big) && \text{(by Lemma A.2)}\\ &= \mathcal{H}'_k(L,M).\end{aligned}$$

The other direction, i.e. $\mathcal{H}'_k(L,M) \subseteq \mathcal{H}_k(L,M)$, can be also proved by induction on $k \in \mathbb{N}$. The base case, when $k = 0$, is again trivial. In the inductive case, we have

$$\begin{aligned}
\mathcal{H}'_k(L,M) &= \mathcal{H}\Big(L, M \cup \mathcal{H}'_{k-1}(L,M)\Big) \\
&\subseteq \mathcal{H}\Big(L, M \cup \mathcal{H}_{k-1}(L,M)\Big) && \text{(by I.H.)} \\
&\subseteq \mathcal{H}\Big(L, M \cup \bigcup_{i \leq k-1} \mathcal{H}_i(L,M)\Big) \\
&= \mathcal{H}_k(L,M). && \square
\end{aligned}$$

Given a hypernet G, by Lemma A.2 and Proposition A.3, there exists a minimum number k such that $G \in \mathcal{H}'_k(L,M)$, which we call the "minimum level" of G.

Lemma A.4. *Any hypernet has a finite number of shallow edges, and a finite number of deep edges.*

Proof. Any hypernet has a finite number of shallow edges by definition. We prove that any hypernet G has a finite number of deep edges, by induction on minimum level k of the hypernet.

When $k = 0$, the hypernet has ho deep edges.

When $k > 0$, each hypernet H that labels a shallow edge of G belongs to $\mathcal{H}'_{k-1}(L,M)$, and therefore its minimum level is less than k. By induction hypothesis, the labelling hypernet H has a finite number of deep edges, and also a finite number of shallow edges. Deep edges of G are given by edges, at any depth, of any hypernet that labels a shallow edge of G. Because there is a finite number of the hypernets that label the shallow edges of G, the number of deep edges of G is finite. $\square$

Appendix B. Plugging

An interfaced labelled monoidal hypergraph can be given by data of the following form: $((V \uplus I \uplus O, E), (S,T), (f_V, f_E))$ where I is the input list, O is the output list, V is the set of all the other vertices, E is the set of edges, (S,T) defines source and target lists, and (f_V, f_E) is labelling functions.

Definition B.1 (Plugging)**.** Let $\mathcal{C}[\vec{\chi^1}, \chi, \vec{\chi^2}] = ((V \uplus I \uplus O, E), (S,T), (f_V, f_E))$ and $\mathcal{C}'[\vec{\chi^3}] = ((V' \uplus I' \uplus O', E'), (S',T'), (f'_V, f'_E))$ be contexts, such that the hole χ and the latter context $\mathcal{C}'$ have the same type and $\vec{\chi^1} \cap \vec{\chi^2} \cap \vec{\chi^3} = \emptyset$. The *plugging* $\mathcal{C}[\vec{\chi^1}, \mathcal{C}', \vec{\chi^2}]$ is a hypernet given by data $((\hat{V}, \hat{E}), (\hat{S}, \hat{T}), (\hat{f}_V, \hat{f}_E))$ such that:

$$\begin{aligned}
\hat{V} &= V \uplus V' \uplus I \uplus O \\
\hat{E} &= (E \backslash \{e_\chi\}) \uplus E' \\
\hat{S}(e) &= \begin{cases} S(e) & (\text{if } e \in E \backslash \{e_\chi\}) \\ g^*(S'(e)) & (\text{if } e \in E') \end{cases} \\
\hat{T}(e) &= \begin{cases} T(e) & (\text{if } e \in E \backslash \{e_\chi\}) \\ g^*(T'(e)) & (\text{if } e \in E') \end{cases}
\end{aligned}$$

$$g(v) = \begin{cases} v & (\text{if } v \in V') \\ (S(e_\chi))_i & (\text{if } v = (I')_i) \\ (T(e_\chi))_i & (\text{if } v = (O')_i) \end{cases}$$

$$\hat{f}_V(v) = \begin{cases} f_V(v) & (\text{if } v \in V) \\ f'_V(v) & (\text{if } v \in V') \end{cases}$$

$$\hat{f}_E(e) = \begin{cases} f_E(e) & (\text{if } e \in E \backslash \{e_\chi\}) \\ f'_E(e) & (\text{if } e \in E') \end{cases}$$

where $e_\chi \in E$ is the hole edge labelled with χ, and $(-)_i$ denotes the i-th element of a list.

In the resulting context $\mathcal{C}[\vec{\chi'}, \mathcal{C}', \vec{\chi''}]$, each edge comes from either $\mathcal{C}$ or $\mathcal{C}'$. If a path in $\mathcal{C}$ does not contain the hole edge e_χ, the path gives a path in $\mathcal{C}[\vec{\chi'}, \mathcal{C}', \vec{\chi''}]$. Conversely, if a path in $\mathcal{C}[\vec{\chi'}, \mathcal{C}', \vec{\chi''}]$ consists of edges from $\mathcal{C}$ only, the path gives a path in $\mathcal{C}$.

Any path in $\mathcal{C}'$ gives a path in $\mathcal{C}[\vec{\chi'}, \mathcal{C}', \vec{\chi''}]$. However, if a path in $\mathcal{C}[\vec{\chi'}, \mathcal{C}', \vec{\chi''}]$ consists of edges from $\mathcal{C}'$ only, the path does not necessarily give a path in $\mathcal{C}'$. The path indeed gives a path in $\mathcal{C}'$, if sources and targets of the hole edge e_χ are distinct in $\mathcal{C}$ (i.e. the hole edge e_χ is not a self-loop).

Appendix C. Rooted states

Lemma C.1. *Let $(X, \twoheadrightarrow)$ is an abstract rewriting system that is deterministic.*

1. *For any $x, y, y' \in X$ such that y and y' are normal forms, and for any $k, h \in \mathbb{N}$, if there exist two sequences $x \twoheadrightarrow^k y$ and $x \twoheadrightarrow^h y'$, then these sequences are exactly the same.*
2. *For any $x, y \in X$ such that y is a normal form, and for any $i, j, k \in \mathbb{N}$ such that $i \neq j$ and $i, j \in \{1, \ldots, k\}$, if there exists a sequence $x \twoheadrightarrow^k y$, then its i-th rewrite $z \twoheadrightarrow z'$ and j-th rewrite $w \twoheadrightarrow w'$ satisfy $z \neq w$.*

Proof. The point (1) is proved by induction on $k + h \in \mathbb{N}$. In the base case, when $k + h = 0$ (i.e. $k = h = 0$), the two sequences are both the empty sequence, and $x = y = y'$. The inductive case, when $k + h > 0$, falls into one of the following two situations. The first situation, where $k = 0$ or $h = 0$, boils down to the base case, because x must be a normal form itself, which means $k = h = 0$. In the second situation, where $k > 0$ and $h > 0$, there exist elements $z, z' \in X$ such that $x \twoheadrightarrow z \twoheadrightarrow^{k-1} y$ and $x \twoheadrightarrow z' \twoheadrightarrow^{h-1} y'$. Because $\twoheadrightarrow$ is deterministic, $z = z'$ follows, and hence by induction hypothesis on $(k-1) + (h-1)$, these two sequences are the same.

The point (2) is proved by contradiction. The sequence $x \twoheadrightarrow^k y$ from x to the normal form y is unique, by the point (1). If its i-th rewrite $z \twoheadrightarrow z'$ and j-th rewrite $w \twoheadrightarrow w'$ satisfy $z = w$, determinism of the system implies that these two rewrites are the same. This means that the sequence $x \twoheadrightarrow^k y$ has a cyclic sub-sequence, and by repeating the cycle different times, one can yield different sequences of rewrites $x \twoheadrightarrow^* y$ from x to y. This contradicts the uniqueness of the original sequence $x \twoheadrightarrow^k y$. □

Lemma C.2. *If a state $\dot{G}$ is rooted, a search sequence $?; |\dot{G}| \overset{\bullet}{\to}^* \dot{G}$ from the initial state $?; |\dot{G}|$ to the state $\dot{G}$ is unique. Moreover, for any i-th search transition and j-th search transition in the sequence such that $i \neq j$, these transitions do not result in the same state.*

Proof. Let X be the set of states with the ?-focus or the ✓-focus. We can define an abstract rewriting system $(X, \twoheadrightarrow)$ of "reverse search" by: $\dot{H} \twoheadrightarrow \dot{H}'$ if $\dot{H}' \xrightarrow{\bullet} \dot{H}$. Any search sequence corresponds to a sequence of rewrites in this rewriting system.

The rewriting system is deterministic, i.e. if $\dot{H}' \xrightarrow{\bullet} \dot{H}$ and $\dot{H}'' \xrightarrow{\bullet} \dot{H}$ then $\dot{H}' = \dot{H}''$, because the inverse $\mapsto^{-1}$ of the interaction rules (Figure 9) is deterministic.

If a search transition changes a focus to the ?-focus, the resulting ?-focus always has an incoming operation edge. This means that, in the rewriting system $(X, \twoheadrightarrow)$, initial states are normal forms. Therefore, by Lemma C.1(1), if there exist two search sequences from the initial state $?; |\dot{G}|$ to the state $\dot{G}$, these search sequences are exactly the same. The rest is a consequence of Lemma C.1(2). □

Lemma C.3. *For any hypernet N, if there exists an operation path from an input to a vertex, the path is unique. Moreover, no edge appears twice in the operation path.*

Proof. Given the hypernet N whose set of (shallow) vertices is X, we can define an abstract rewriting system $(X, \twoheadrightarrow)$ of "reverse connection" by: $v \twoheadrightarrow v'$ if there exists an operation edge whose unique source is v' and targets include v. Any operation path from an input to a vertex in N corresponds to a sequence of rewrites in this rewriting system.

This rewriting system is deterministic, because each vertex can have at most one incoming edge in a hypergraph (Definition 3.2) and each operation edge has exactly one source. Because inputs of the hypernet N have no incoming edges, they are normal forms in this rewriting system. Therefore, by Lemma C.1(1), an operation path from any input to any vertex is unique.

The rest is proved by contradiction. We assume that, in an operation path P from an input to a vertex, the same operation edge e appears twice. The edge e has one source, which either is an input of the hypernet N or has an incoming edge. In the former case, the edge e can only appear as the first edge of the operation path P, which is a contradiction. In the latter case, the operation edge e has exactly one incoming edge e' in the hypernet N. In the operation path P, each appearance of the operation edge e must be preceded by this edge e' via the same vertex. This contradicts Lemma C.1(2). □

Lemma C.4. *For any rooted state $\dot{G}$, if its focus source (i.e. the source of the focus) does not coincide with the unique input, then there exists an operation path from the input to the focus source.*

Proof. By Lemma C.2, the rooted state $\dot{G}$ has a unique search sequence $?; |\dot{G}| \xrightarrow{\bullet}{}^{*} \dot{G}$. The proof is by the length k of this sequence.

In the base case, where $k = 0$, the state $\dot{G}$ itself is an initial state, which means the input and focus source coincide in $\dot{G}$.

In the inductive case, where $k > 0$, there exists a state $\dot{G}'$ such that $?; |\dot{G}| \xrightarrow{\bullet}{}^{k-1} \dot{G}' \xrightarrow{\bullet} \dot{G}$. The proof here is by case analysis on the interaction rule used in $\dot{G}' \xrightarrow{\bullet} \dot{G}$.

- When the interaction rules in Figure 18a, 18b, 18c, 9e are used, the transition $\dot{G}' \xrightarrow{\bullet} \dot{G}$ only changes a focus label.
- When the interaction rule in Figure 9b is used, the transition $\dot{G}' \xrightarrow{\bullet} \dot{G}$ turns the focus and its outgoing operation edge $e_{G'}$ into an operation edge e_G and its outgoing focus. By induction hypothesis on $\dot{G}'$, the focus source coincides with its input, or there exists an operation path from the input to the focus source, in $\dot{G}'$.

  In the former case, in $\dot{G}$, the source of the operation edge e_G coincides with the input. The edge e_G itself gives the desired operation path in $\dot{G}$.

In the latter case, the operation path $P_{G'}$ from the input to the focus source in $\dot{G}'$ does not contain the outgoing operation edge $e_{G'}$ of the focus; otherwise, the edge $e_{G'}$ must be preceded by the focus edge in the operation path $P_{G'}$, which is a contradiction. Therefore, the operation path $P_{G'}$ in $\dot{G}'$ is inherited in $\dot{G}$, becoming a path P_G from the input to the source of the incoming operation edge e_G of the focus. In the state $\dot{G}$, the path P_G followed by the edge e_G yields the desired operation path.

- When the interaction rule in Figure 9c is used, the transition $\dot{G}' \xrightarrow{\bullet} \dot{G}$ changes the focus from a $(k+1)$-th outgoing edge of an operation edge e to a $(k+2)$-th outgoing edge of the same operation edge e, for some $k \in \mathbb{N}$. In $\dot{G}'$, the focus source is not an input, and therefore, there exists an operation path $P_{G'}$ from the input to the focus source, by induction hypothesis.

  The operation path $P_{G'}$ ends with the operation edge e, and no outgoing edge of the edge e is involved in the path $P_{G'}$; otherwise, the edge e must appear more than once in the path $P_{G'}$, which is a contradiction by Lemma C.3. Therefore, the path $P_{G'}$ is inherited exactly as it is in $\dot{G}$, and it gives the desired operation path.
- When the interaction rule in Figure 9d is used, by the same reasoning as in the case of Figure 9c, $\dot{G}'$ has an operation path $P_{G'}$ from the input to the focus source, where the incoming operation edge $e_{G'}$ of the focus appears exactly once, at the end. Removing the edge $e_{G'}$ from the path $P_{G'}$ yields another operation path P from the input in $\dot{G}'$, and it also gives an operation path from the input to the focus source in $\dot{G}$. □

Lemma C.5. *For any state $\dot{G}$ with a $\mathtt{t}$-focus such that $\mathtt{t} \neq ?$, if $\dot{G}$ is rooted, then there exists a search sequence $?; |\dot{G}| \xrightarrow{\bullet}{}^{*} \langle \dot{G} \rangle_{?/\mathtt{t}} \xrightarrow{\bullet}{}^{+} \dot{G}$.*

Proof. By Lemma C.2, the rooted state $\dot{G}$ has a unique search sequence $?; |\dot{G}| \xrightarrow{\bullet}{}^{*} \dot{G}$. The proof is to show that a transition from the state $\langle \dot{G} \rangle_{?/\mathtt{t}}$ appears in this search sequence, and it is by the length k of the search sequence.

Because $\dot{G}$ does not have the ?-focus, $k = 0$ is impossible, and therefore the base case is when $k = 1$. The search transition $?; |\dot{G}| \xrightarrow{\bullet} \dot{G}$ must use one of the interaction rules in Figure 18a, 18b, 18c, 9e. This means $?; |\dot{G}| = \langle \dot{G} \rangle_{?/\mathtt{t}}$.

In the inductive case, where $k > 0$, there exists a state $\dot{G}'$ such that $?; |\dot{G}| \xrightarrow{\bullet}{}^{k-1} \dot{G}' \xrightarrow{\bullet} \dot{G}$. The proof here is by case analysis on the interaction rule used in $\dot{G}' \xrightarrow{\bullet} \dot{G}$.

- When the interaction rule in Figure 18a, 18b, 18c, 9e is used, $?; |\dot{G}| = \langle \dot{G} \rangle_{?/\mathtt{t}}$.
- Because $\dot{G}$ does not have the ?-focus, the interaction rules in Figure 9b, 9c can be never used in $\dot{G}' \xrightarrow{\bullet} \dot{G}$.
- When the interaction rule in Figure 9d is used, $\dot{G}'$ has the ✓-focus, which is a $(k+1)$-th outgoing edge of an operation edge e, for some $k \in \mathbb{N}$. The operation edge e becomes the outgoing edge of the focus in $\dot{G}$. By induction hypothesis on $\dot{G}'$, we have

$$?; |\dot{G}| \xrightarrow{\bullet}{}^{*} \langle \dot{G}' \rangle_{?/\checkmark} \xrightarrow{\bullet}{}^{+} \dot{G}' \xrightarrow{\bullet} \dot{G}. \tag{A}$$

  If $k = 0$, in $\dot{G}'$, the focus is the only outgoing edge of the operation edge e. Because $\langle \dot{G}' \rangle_{?/\checkmark}$ is not an initial state, it must be a result of the interaction rule in Figure 9b, which means the search sequence (A) is factored through as:

$$?; |\dot{G}| \xrightarrow{\bullet}{}^{*} \langle \dot{G} \rangle_{?/\mathtt{t}} \xrightarrow{\bullet} \langle \dot{G}' \rangle_{?/\checkmark} \xrightarrow{\bullet}{}^{+} \dot{G}' \xrightarrow{\bullet} \dot{G}.$$

  If $k > 0$, for each $m \in \{0, \ldots, k\}$, let $\dot{N}_m$ be a state with the ?-focus, such that $|\dot{N}_m| = |\dot{G}'|$ and the focus is an $(m+1)$-th outgoing edge of the operation edge e. This

means $\dot{N}_k = \langle \dot{G}' \rangle_{?/\checkmark}$. The proof concludes by combining the following internal lemma with (A), taking k as m. □

Lemma C.6. *For any $m \in \{0, \ldots, k\}$, if there exists $h < k$ such that $?; |\dot{G}| \overset{\bullet}{\to}^{h} \dot{N}_m$, then it is factored through as $?; |\dot{G}| \overset{\bullet}{\to}^{*} \langle \dot{G} \rangle_{?/\mathtt{t}} \overset{\bullet}{\to}^{+} \dot{N}_m$.*

Proof. By induction on m. In the base case, when $m = 0$, the focus of $\dot{N}_m$ is the first outgoing edge of the operation edge e. This state is not initial, and therefore must be a result of the interaction rule in Figure 9b, which means

$$?; |\dot{G}| \overset{\bullet}{\to}^{*} \langle \dot{G} \rangle_{?/\mathtt{t}} \overset{\bullet}{\to} \dot{N}_m.$$

In the inductive case, when $m > 0$, the state $\dot{N}_m$ is not an initial state and must be a result of the interaction rule in Figure 9c, which means

$$?; |\dot{G}| \overset{\bullet}{\to}^{*} \langle \dot{N_{m-1}} \rangle_{\checkmark/?} \overset{\bullet}{\to} \dot{N}_m.$$

The first half of this search sequence, namely $?; |\dot{G}| \overset{\bullet}{\to}^{*} \langle \dot{N_{m-1}} \rangle_{\checkmark/?}$, consists of $h - 1 < k$ transitions. Therefore, by (outer) induction hypothesis on $h - 1$, we have

$$?; |\dot{G}| \overset{\bullet}{\to}^{*} \dot{N_{m-1}} \overset{\bullet}{\to}^{+} \langle \dot{N_{m-1}} \rangle_{\checkmark/?} \overset{\bullet}{\to} \dot{N}_m.$$

The first part, namely $?; |\dot{G}| \overset{\bullet}{\to}^{*} \dot{N_{m-1}}$, consists of less than k transitions. Therefore, by (inner) induction hypothesis on $m - 1$, we have

$$?; |\dot{G}| \overset{\bullet}{\to}^{*} \langle \dot{G} \rangle_{?/\mathtt{t}} \overset{\bullet}{\to}^{+} \dot{N_{m-1}} \overset{\bullet}{\to}^{+} \langle \dot{N_{m-1}} \rangle_{\checkmark/?} \overset{\bullet}{\to} \dot{N}_m.$$ □

Lemma C.7.

(1) *For any state $\dot{N}$, if it has a path to the focus source that is not an operation path, then it is not rooted.*
(2) *For any focus-free hypernet H and any focussed context $\dot{\mathcal{C}}[\chi]$ with one hole edge, such that $\dot{\mathcal{C}}[H]$ is a state, if the hypernet H is one-way and the context $\dot{\mathcal{C}}$ has a path to the focus source that is not an operation path, then the state $\dot{\mathcal{C}}[H]$ is not rooted.*
(3) *For any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\vec{\chi}]; \vec{G}; \vec{H})$ of an output-closed pre-template $\lhd$, if the context $\dot{\mathcal{C}}[\vec{\chi}]$ has a path to the focus source that is not an operation path, then at least one of the states $\dot{\mathcal{C}}[\vec{G}]$ and $\dot{\mathcal{C}}[\vec{H}]$ is not rooted.*

Proof of the point (1). Let P be the path in $\dot{N}$ to the focus source that is not an operation path. The proof is by contradiction; we assume that $\dot{N}$ is a rooted state.

Because of P, the focus source is not an input. Therefore by Lemma C.4, the state $\dot{N}$ has an operation path from its unique input to the focus source. This operation path contradicts the path P, which is not an operation path, because each operation edge has only one source and each vertex has at most one incoming edge. □

Proof of the point (2). Let P be the path in $\dot{\mathcal{C}}$ to the focus source that is not an operation path.

If the path P contains no hole edge, it gives a path in the state $\dot{\mathcal{C}}[H]$ to the focus source that is not an operation path. By the point (1), the state is not rooted.

Otherwise, i.e. if the path P contains a hole edge, we give a proof by contradiction; we assume that the state $\dot{\mathcal{C}}[H]$ is rooted. We can take a suffix of the path P, so that it gives a path from a target of a hole edge to the focus source in $\dot{\mathcal{C}}$, and moreover, gives a path P' from a source of an edge from H to the focus source in $\dot{\mathcal{C}}[H]$. This implies the focus source

is not an input, and therefore by Lemma C.4, the state $\dot{\mathcal{C}}[H]$ has an operation path from its unique input to the focus source. This operation path must have P' has a suffix, meaning P' is also an operation path, because each operation edge has only one source and each vertex has at most one incoming edge. Moreover, H must have an operation path from an input to an output, such that the input and the output have type $\star$ and the path ends with the first edge of the path P'. This contradicts H being one-way. □

Proof of the point (3). Let P be the path in $\dot{\mathcal{C}}$ to the focus source that is not an operation path.

If the path P contains no hole edge, it gives a path in the states $\dot{\mathcal{C}}[\vec{G}]$ and $\dot{\mathcal{C}}[\vec{H}]$ to the focus source that is not an operation path. By the point (1), the states are not rooted.

Otherwise, i.e. if the path P contains a hole edge, we can take a suffix of P that gives a path P' from a source of a hole edge e to the focus source in $\dot{\mathcal{C}}$, so that the path P' does not contain any hole edge. We can assume that the hole edge e is labelled with χ_1, without loss of generality. The path P' gives paths P'_G and P'_H to the focus source, in contexts $\dot{\mathcal{C}}[\chi_1, \vec{G}\backslash\{G_1\}]$ and $\dot{\mathcal{C}}[\chi_1, \vec{H}\backslash\{H_1\}]$, respectively. The paths P'_G and P'_H are not an operation path, because they start with the hole edge e labelled with χ_1.

Because $\lhd$ is output-closed, G_1 or H_1 is one-way. By the point (2), at least one of the states $\dot{\mathcal{C}}[\vec{G}]$ and $\dot{\mathcal{C}}[\vec{H}]$ is not rooted. □

Lemma C.8. *If a rewrite transition $\dot{G} \to \dot{G}'$ is stationary, it preserves the rooted property, i.e. $\dot{G}$ being rooted implies $\dot{G}'$ is also rooted.*

Proof. The stationary rewrite transition $\dot{G} \to \dot{G}'$ is in the form of $\mathcal{C}[\lightning ;_i H] \to \mathcal{C}[?;_i H']$, where $\mathcal{C}$ is a focus-free simple context, H is a focus-free one-way hypernet, H' is a focus-free hypernet and $i \in \mathbb{N}$. We assume $\mathcal{C}[\lightning ;_i H]$ is rooted, and prove that $\mathcal{C}[?;_i H']$ is rooted, i.e. $?;\mathcal{C}[H'] \xrightarrow{\bullet}{}^{*} \mathcal{C}[?;_i H']$. By Lemma C.5, there exists a number $k \in \mathbb{N}$ such that:

$$?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{k} \mathcal{C}[?;_i H] \xrightarrow{\bullet}{}^{+} \mathcal{C}[\lightning ;_i H].$$

The rest of the proof is by case analysis on the number k.

- When $k = 0$, i.e. $?;\mathcal{C}[H] = \mathcal{C}[?;_i H]$, the unique input and the i-th source of the hole coincide in the simple context $\mathcal{C}$. Therefore, $?;\mathcal{C}[H'] = \mathcal{C}[?;_i H']$, which means $\mathcal{C}[?;_i H']$ is rooted.
- When $k > 0$, there exists a state $\dot{N}$ such that $?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{k-1} \dot{N} \xrightarrow{\bullet} \mathcal{C}[?;_i H]$. By the following internal lemma (Lemma C.9), there exists a focussed simple context $\dot{\mathcal{C}_N}$, whose focus is not entering nor exiting, and we have two search sequences:

  $$?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{k-1} \dot{\mathcal{C}_N}[H] \xrightarrow{\bullet} \mathcal{C}[?;_i H],$$
  $$?;\mathcal{C}[H'] \xrightarrow{\bullet}{}^{k-1} \dot{\mathcal{C}_N}[H'].$$

  The last search transition $\dot{\mathcal{C}_N}[H] \xrightarrow{\bullet} \mathcal{C}[?;_i H]$, which yields the ?-focus, must use the interaction rule in Figure 9b, 9c. Because the focus is not entering nor exiting in the simple context $\dot{\mathcal{C}_N}$, either of the two interaction rules acts on the focus and an edge of the context. This means that the same interaction is possible in the state $\dot{\mathcal{C}_N}[H']$, yielding:

  $$?;\mathcal{C}[H'] \xrightarrow{\bullet}{}^{k-1} \dot{\mathcal{C}_N}[H'] \xrightarrow{\bullet} \mathcal{C}[?;_i H'],$$

  which means $\mathcal{C}[?;_i H']$ is rooted. □

Lemma C.9. *For any $m \in \{0, \ldots, k-1\}$ and any state $\dot{N}$ such that $?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{m} \dot{N} \xrightarrow{\bullet}{}^{k-m} \mathcal{C}[?;_i H]$, the following holds.*

(A) If there exists a focussed simple context $\dot{\mathcal{C}_N}$ such that $\dot{N} = \dot{\mathcal{C}_N}[H]$, the focus of the context $\dot{\mathcal{C}_N}$ is not entering.

(B) If there exists a focussed simple context $\dot{\mathcal{C}_N}$ such that $\dot{N} = \dot{\mathcal{C}_N}[H]$, the focus of the context $\dot{\mathcal{C}_N}$ is not exiting.

(C) There exists a focussed simple context $\dot{\mathcal{C}_N}$ such that $\dot{N} = \dot{\mathcal{C}_N}[H]$, and $?;\mathcal{C}[H'] \xrightarrow{\bullet}{}^{m} \dot{\mathcal{C}_N}[H']$ holds.

Proof. Firstly, because search transitions do not change an underlying hypernet, if there exists a focussed simple context $\dot{\mathcal{C}_N}$ such that $\dot{N} = \dot{\mathcal{C}_N}[H]$, $|\dot{\mathcal{C}_N}| = \mathcal{C}$ necessarily holds.

The point (A) is proved by contradiction; we assume that the context $\dot{\mathcal{C}_N}$ has an entering focus. This means that there exist a number $p \in \mathbb{N}$ and a focus label $\mathtt{t} \in \{?, \checkmark, \lightning\}$ such that $\dot{\mathcal{C}_N} = \mathcal{C}[\mathtt{t};_p H]$. By Lemma C.5, there exists a number h such that $h \leq m$ and:

$$?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{h} \mathcal{C}[?;_p H] \xrightarrow{\bullet}{}^{k-h} \mathcal{C}[?;_i H]. \tag{\$}$$

We derive a contradiction by case analysis on the numbers p and h.

- If $p = i$ and $h = 0$, the state $\mathcal{C}[?;_i H]$ must be initial, but it is a result of a search transition because $k - h > 0$. This is a contradiction.
- If $p = i$ and $h > 0$, two different transitions in the search sequence (\$) result in the same state, because of $h > 0$ and $k - h > 0$, which contradicts Lemma C.2.
- If $p \neq i$, by Definition 7.4, there exists a state $\dot{N'}$ with the $\lightning$-focus such that $\mathcal{C}[?;_p H] \xrightarrow{\bullet} \dot{N'}$. This contradicts the search sequence (\$), because $k - h > 0$ and search transitions are deterministic.

The point (B) follows from the contraposition of Lemma C.7(2), because H is one-way and $\dot{N}$ is rooted. The rooted property of $\dot{N}$ follows from the fact that search transitions do not change underlying hypernets.

The point (C) is proved by induction on $m \in \{0, \ldots, k-1\}$. In the base case, when $m = 0$, we have $?;\mathcal{C}[H] = \dot{N}$, and therefore the context $?;\mathcal{C}$ can be taken as $\dot{\mathcal{C}_N}$. This means $?;\mathcal{C}[H'] = \dot{\mathcal{C}_N}[H']$.

In the inductive case, when $m > 0$, there exists a state $\dot{N'}$ such that

$$?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{m-1} \dot{N'} \xrightarrow{\bullet} \dot{N} \xrightarrow{\bullet}{}^{k-m} \mathcal{C}[?;_i H].$$

By the induction hypothesis, there exists a focussed simple context $\dot{\mathcal{C}_{N'}}$ such that $\dot{N'} = \dot{\mathcal{C}_{N'}}[H]$ and

$$?;\mathcal{C}[H] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}_{N'}}[H] \xrightarrow{\bullet} \dot{N} \xrightarrow{\bullet}{}^{k-m} \mathcal{C}[?;_i H],$$
$$?;\mathcal{C}[H'] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}_{N'}}[H'].$$

Our goal here is to find a focussed simple context $\dot{\mathcal{C}_N}$, such that $\dot{N} = \dot{\mathcal{C}_N}[H]$ and $\dot{\mathcal{C}_{N'}}[H'] \xrightarrow{\bullet} \dot{\mathcal{C}_N}[H']$.

In the search transition $\dot{\mathcal{C}_{N'}}[H] \xrightarrow{\bullet} \dot{N}$, the only change happens to the focus and its incoming or outgoing edge e in the state $\dot{\mathcal{C}_{N'}}[H]$. By the points (A) and (B), the focus is not entering nor exiting in the context $\dot{\mathcal{C}_{N'}}$, which means the edge e must be from the context, not from H.

Now that no edge from H is changed in $\dot{\mathcal{C}_{N'}}[H] \xrightarrow{\bullet} \dot{N}$, there exists a focussed simple context $\dot{\mathcal{C}_N}$ such that $\dot{N} = \dot{\mathcal{C}_N}[H]$, and moreover, $\dot{\mathcal{C}_{N'}}[H'] \xrightarrow{\bullet} \dot{\mathcal{C}_N}[H']$. □

Appendix D. Stable hypernets

Definition D.1 (Accessible paths)**.**

- A path of a hypernet is said to be *accessible* if it consists of edges whose all sources have type $\star$.
- An accessible path is called *stable* if the labels of its edges are included in $\{\mathsf{I}\} \cup \mathbb{O}_{\checkmark}$.
- An accessible path is called *active* if it starts with one active operation edge and possibly followed by a stable path.

A stable hypernet always has at least one edge, and any non-output vertex is labelled with $\star$. It has a tree-like shape.

Lemma D.2 (Shape of stable hypernets)**.** (1) *In any stable hypernet, if a vertex v' is reachable from another vertex v such that $v \neq v'$, there exists a unique path from the vertex v to the vertex v'.*
(2) *Any stable hypernet has no cyclic path, i.e. a path from a vertex to itself.*
(3) *Let $\mathcal{C} : \star \Rightarrow \otimes_{i=1}^{m} \ell_i$ be a simple context such that: its hole has one source and at least one outgoing edge; and its unique input is the hole's source. There are no two stable hypernets G and G' that satisfy $G = \mathcal{C}[G']$.*

Proof. To prove the point (1), assume there are two different paths from the vertex v to the vertex v'. These paths, i.e. non-empty sequences of edges, have to involve an edge with more than one source, or two different edges that share the same target. However, neither of these is possible in a stable hypernet, because both a passive operation edge and an instance edge have only one source and vertices can have at most one incoming edge. The point (1) follows from this by contradiction.

If a stable hypernet has a cyclic path from a vertex v to itself, there must be infinitely many paths from the input to the vertex v, depending on how many times the cycle is included. This contradicts the point (1).

The point (3) is also proved by contradiction. Assume that there exist two stable hypernets G and G' that satisfy $G = \mathcal{C}[G']$ for the simple context $\mathcal{C}$. In the stable hypernet G, a vertex is always labelled with $\star$ if it is not an output. However, in the simple context $\mathcal{C}$, there exists at least one target of the hole that is not an output of the context but not labelled with $\star$ either. This contradicts $\mathcal{C}[G']$ being a stable hypernet. □

Lemma D.3. *For any state $\dot{N}$, and its vertex v, such that the vertex v is not a target of an instance edge or a passive operation edge, if an accessible path from the vertex v is stable or active, then the path has no multiple occurrences of a single edge.*

Proof. Any stable or active path consists of edges that has only one source. As a consequence, except for the first edge, no edge appears twice in the stable path. If the stable path is from the vertex v, its first edge also does not appear twice, because v is not a target of an instance edge or a passive operation edge. □

Lemma D.4. *For any state $\dot{N}$, and its vertex v, such that the vertex v is not a target of an instance edge or a passive operation edge, the following are equivalent.*

(A) There exist a focussed simple context $\dot{\mathcal{C}}[\chi]$ and a stable hypernet G, such that $\dot{N} = \dot{\mathcal{C}}[G]$, where the vertex v of $\dot{N}$ corresponds to a unique source of the hole edge in $\dot{\mathcal{C}}$.

(B) Any accessible path from the vertex v in $\dot{N}$ is a stable path.

Proof of (A) ⇒ (B). Because no output of a stable hypernet has type $\star$, any path from the vertex v in $\dot{\mathcal{C}}[G]$ gives a path from the unique input in G. In the stable hypernet G, any path from the unique input is a stable path. □

Proof of (B) ⇒ (A). In the state $\dot{N}$, the focus target has to be a source of an edge, which forms an accessible path itself. By Lemma D.3, in the state $\dot{N}$, we can take maximal stable paths from the vertex v, in the sense that appending any edge to these paths, if possible, does not give a stable path.

If any of these maximal stable paths is to some vertex, the vertex does not have type $\star$; this can be confirmed as follows. If the vertex has type $\star$, it is not an output, so it is a source of an instance, focus, operation or contraction edge. The case of an instance or passive operation edge contradicts the maximality. The other case yields a non-stable accessible path that contradicts the assumption (B).

Collecting all edges contained by the maximal stable paths, therefore, gives the desired hypernet G. These edges are necessarily all shallow, because of the vertex v of $\dot{N}$. The focussed context $\dot{\mathcal{C}}[\chi]$, whose hole is shallow, can be made of all the other edges (at any depth) of the state $\dot{N}$. □

Lemma D.5. *Let $\dot{N}$ be a state, where the focus is an incoming edge of an operation edge e, whose label ϕ takes at least one eager arguments. Let k denote the number of eager arguments of ϕ.*

For each $i \in \{1, \ldots, k\}$, let $sw_i(\dot{N})$ be a state such that: both states $sw_i(\dot{N})$ and $\dot{N}$ have the same focus label and the same underlying hypernet, and the focus in $sw_i(\dot{N})$ is the i-th outgoing edge of the operation edge e.

For each $i \in \{1, \ldots, k\}$, the following are equivalent.

(A) In $\dot{N}$, any accessible path from an i-th target of the operation edge e is a stable (resp. active) path.

(B) In $sw_i(\dot{N})$, any accessible path from the focus target is a stable (resp. active) path.

Proof. The only difference between $\dot{N}$ and $sw_i(\dot{N})$ is the swap of the focus with the operation edge e, and these two edges form an accessible path in the states $\dot{N}$ and $sw_i(\dot{N})$, individually or together (in an appropriate order). Therefore, there is one-to-one correspondence between accessible paths from an i-th target of the edge e in $\dot{N}$, and accessible paths from the focus target in $sw_i(\dot{N})$.

When (A) is the case, in $\dot{N}$, any accessible paths from an i-th target of the edge e does not contain the focus nor the edge e; otherwise there would be an accessible path that contains the focus and hence not stable nor active, which is a contradiction. This means that, in $sw_i(\dot{N})$, any accessible path from the focus target also does not contain the focus nor the edge e, and the path must be a stable (resp. active) path.

When (B) is the case, the proof takes the same reasoning in the reverse way. □

Lemma D.6. *Let $\dot{N}$ be a rooted state with the ?-focus, such that the focus is not an incoming edge of a contraction edge.*

(1) *$\dot{N} \xrightarrow{\bullet}{}^{+} \langle \dot{N} \rangle_{\checkmark/?}$, if and only if any accessible path from the focus target in $\dot{N}$ is a stable path.*
(2) *$\dot{N} \xrightarrow{\bullet}{}^{+} \langle \dot{N} \rangle_{\lightning/?}$, if and only if any accessible path from the focus target in $\dot{N}$ is an active path.*

Proof of the forward direction. Let $\mathtt{t}$ be either '$\checkmark$' or '$\lightning$'. The assumption is $\dot{N} \xrightarrow{\bullet}{}^{*} \langle \dot{N} \rangle_{\mathtt{t}/?}$. We prove the following, by induction on the length n of this search sequence:

- any accessible path from the focus target in $\dot{N}$ is a stable path, when $\mathtt{t} = \checkmark$, and
- any accessible path from the focus target in $\dot{N}$ is an active path, when $\mathtt{t} = \lightning$.

In the base case, where $n = 1$, because the focus is not an incoming edge of a contraction edge, the focus target is a source of an instance edge, or an operation edge labelled with $\phi \in \mathbb{O}_{\mathtt{t}}$ that takes no eager argument. In either situation, the outgoing edge of the focus gives the only possible accessible path from the focus target. The path is stable when $\mathtt{t} = \checkmark$, and active when $\mathtt{t} = \lightning$.

In the inductive case, where $n > 1$, the focus target is a source of an operation edge e_ϕ labelled with an operation $\phi \in \mathbb{O}_{\mathtt{t}}$ that takes at least one eager argument.

Let k denote the number of eager arguments of $\phi_{\mathtt{t}}$, and i be an arbitrary number in $\{1, \ldots, k\}$. Let $sw_i(\dot{N})$ be the state as defined in Lemma D.5. Because $\dot{N}$ is rooted, by Lemma C.5, the given search sequence gives the following search sequence (proof by induction on $k - i$):

$$?; |\dot{N}| \xrightarrow{\bullet}{}^{*} \dot{N} \xrightarrow{\bullet}{}^{+} sw_i(\dot{N}) \xrightarrow{\bullet}{}^{+} \langle sw_i(\dot{N}) \rangle_{\checkmark/?} \xrightarrow{\bullet}{}^{+} \langle \dot{N} \rangle_{\mathtt{t}/?}.$$

By induction hypothesis on the intermediate sequence $sw_i(\dot{N}) \xrightarrow{\bullet}{}^{+} \langle sw_i(\dot{N}) \rangle_{\checkmark/?}$, any accessible path from the focus target in $sw_i(\dot{N})$ is a stable path. By Lemma D.5, any accessible path from an i-th target of the operation edge e_ϕ in $\dot{N}$ is a stable path.

In $\dot{N}$, any accessible path from the focus target is given by the operation edge e_ϕ followed by an accessible path, which is proved to be stable above, from a target of e_ϕ. Any accessible path from the focus target is therefore stable when $\mathtt{t} = \checkmark$, and active when $\mathtt{t} = \lightning$. □

Proof of the backward direction. Let $\mathtt{t}$ be either '$\checkmark$' or '$\lightning$'. The assumption is the following:

- any accessible path from the focus target in $\dot{N}$ is a stable path, when $\mathtt{t} = \checkmark$, and
- any accessible path from the focus target in $\dot{N}$ is an active path, when $\mathtt{t} = \lightning$.

Our goal is to show $\dot{N} \xrightarrow{\bullet}{}^{*} \langle \dot{N} \rangle_{\mathtt{t}/?}$.

In the state $\dot{N}$, the focus target has to be a source of an edge, which forms an accessible path itself. By Lemma D.3, we can define $r(\dot{N})$ by the maximum length of stable paths from the focus target. This number $r(\dot{N})$ is well-defined and positive. We prove $\dot{N} \xrightarrow{\bullet}{}^{*} \langle \dot{N} \rangle_{\mathtt{t}/?}$ by induction on $r(\dot{N})$.

In the base case, where $r(\dot{N}) = 1$, the outgoing edge of the focus is the only possible accessible path from the focus target. The outgoing edge is not a contraction edge by the assumption, and hence it is an instance edge, or an operation edge labelled with $\phi \in \mathbb{O}_{\mathtt{t}}$ that takes no eager argument. We have $\dot{N} \xrightarrow{\bullet} \langle \dot{N} \rangle_{\mathtt{t}/?}$.

In the inductive case, where $r(\dot{N}) > 1$, the outgoing edge of the focus is an operation edge e_ϕ labelled with $\phi \in \mathbb{O}_{\mathtt{t}}$ that takes at least one eager argument. Any accessible path from the focus target in $\dot{N}$ is given by the edge e_ϕ followed by a stable path from a target of e_ϕ.

Let k denote the number of eager arguments of $\phi_{\mathtt{t}}$, and i be an arbitrary number in $\{1, \ldots, k\}$. Let $sw_i(\dot{N})$ be the state as defined in Lemma D.5.

By the assumption, any accessible path from an i-th target of the operation edge e_ϕ in $\dot{N}$ is a stable path. Therefore by Lemma D.5, in $sw_i(\dot{N})$, any accessible path from the focus target is a stable path. Moreover, these paths in $\dot{N}$ and $sw_i(\dot{N})$ correspond to each other. By Lemma D.3, we can define $r(sw_i(\dot{N}))$ by the maximum length of stable paths from the

focus target. This number $r(sw_i(\dot{N}))$ is well-defined, and satisfies $r(sw_i(\dot{N})) < r(\dot{N})$. By induction hypothesis on this number, we have:

$$sw_i(\dot{N}) \xrightarrow{\bullet}{}^{*} \langle sw_i(\dot{N})\rangle_{\checkmark/?}.$$

Combining this search sequence with the following possible search transitions concludes the proof:

$$\dot{N} \xrightarrow{\bullet} sw_1(\dot{N}),$$

$$\langle sw_i(\dot{N})\rangle_{\checkmark/?} \xrightarrow{\bullet} sw_{i+1}(\dot{N}), \qquad \text{(when } k \neq 1 \text{ and } i < k\text{)}$$

$$\langle sw_k(\dot{N})\rangle_{\checkmark/?} \xrightarrow{\bullet} \langle \dot{N}\rangle_{\mathtt{t}/?}. \qquad \square$$

Appendix E. Parameterised (contextual) refinement and equivalence

Lemma E.1. *For any focus-free contexts $\mathcal{C}_1[\vec{\chi'}, \chi, \vec{\chi''}]$ and $\mathcal{C}_2$ such that $\mathcal{C}_1[\vec{\chi'}, \mathcal{C}_2, \vec{\chi''}]$ is defined, if both $\mathcal{C}_1$ and $\mathcal{C}_2$ are binding-free, then $\mathcal{C}_1[\vec{\chi'}, \mathcal{C}_2, \vec{\chi''}]$ is also binding-free.*

Proof. Let $\mathcal{C}$ denote $\mathcal{C}_1[\vec{\chi'}, \mathcal{C}_2, \vec{\chi''}]$, and e_χ denote the hole edge of $\mathcal{C}_1$ labelled with χ.

The proof is by contradiction. We assume that there exists a path P in $\mathcal{C}$, from a source of a contraction, atom, box or hole edge e, to a source of a hole edge e'. We derive a contradiction by case analysis on the path P.

- When e' comes from $\mathcal{C}_1$, and the path P consists of edges from $\mathcal{C}_1$ only, the path P gives a path in $\mathcal{C}_1$ that contradicts $\mathcal{C}_1$ being binding-free.
- When e' comes from $\mathcal{C}_1$, and the path P contains an edge from $\mathcal{C}_2$, by finding the last edge from $\mathcal{C}_2$ in P, we can take a suffix of P that gives a path from a target of the hole edge e_χ to a source of a hole edge, in $\mathcal{C}_1$. Adding the hole edge e_χ at the beginning yields a path in $\mathcal{C}_1$ that contradicts $\mathcal{C}_1$ being binding-free.
- When both e and e' come from $\mathcal{C}_2$, and the path P gives a path in $\mathcal{C}_2$, this contradicts $\mathcal{C}_2$ being binding-free.
- When both e and e' come from $\mathcal{C}_2$, and the path P does not give a single path in $\mathcal{C}_2$, there exists a path from a source of the hole edge e_χ to a source of the hole edge e_χ, in $\mathcal{C}_1$. This path contradicts $\mathcal{C}_1$ being binding-free.
- When e comes from $\mathcal{C}_1$ and e' comes from $\mathcal{C}_2$, by finding the first edge from $\mathcal{C}_2$ in P, we can take a prefix of P that gives a path from a source of a contraction, atom, box or hole edge to a source of the hole edge e_χ, in $\mathcal{C}_1$. This path contradicts $\mathcal{C}_1$ being binding-free. $\square$

Lemma E.2. *For any set $\mathbb{C}$ of contexts that is closed under plugging, and any preorder Q on natural numbers, the following holds.*

- *$\dot{\preceq}_Q$ and $\preceq_Q^{\mathbb{C}}$ are reflexive.*
- *$\dot{\preceq}_Q$ and $\preceq_Q^{\mathbb{C}}$ are transitive.*
- *$\dot{\simeq}_Q$ and $\simeq_Q^{\mathbb{C}}$ are equivalences.*

Proof. Because $\dot{\simeq}_Q$ and $\simeq_Q^{\mathbb{C}}$ are defined as a symmetric subset of $\dot{\preceq}_Q$ and $\preceq_Q^{\mathbb{C}}$, respectively, $\dot{\simeq}_Q$ and $\simeq_Q^{\mathbb{C}}$ are equivalences if $\dot{\preceq}_Q$ and $\preceq_Q^{\mathbb{C}}$ are preorders.

Reflexivity and transitivity of $\dot{\preceq}_Q$ is a direct consequence of those of the preorder Q.

For any focus-free hypernet H, and any focus-free context $\mathcal{C}[\chi] \in \mathbb{C}$ such that $?;\mathcal{C}[H]$ is a state, $?;\mathcal{C}[H] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H]$ because of reflexivity of $\dot{\preceq}_Q$.

For any focus-free hypernets H_1, H_2 and H_3, and any focus-free context $\mathcal{C}[\chi] \in \mathbb{C}$, such that $H_1 \preceq_Q^{\mathbb{C}} H_2$, $H_2 \preceq_Q^{\mathbb{C}} H_3$, and both $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_3]$ are states, our goal is to show $?;\mathcal{C}[H_1] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H_3]$. Because $H_1 \preceq_Q^{\mathbb{C}} H_2$ and $H_2 \preceq_Q^{\mathbb{C}} H_3$, all three hypernets H_1, H_2 and H_3 have the same type, and hence $?;\mathcal{C}[H_2]$ is also a state. Therefore, we have $?;\mathcal{C}[H_1] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H_2]$ and $?;\mathcal{C}[H_2] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H_3]$, and the transitivity of $\dot{\preceq}_Q$ implies $?;\mathcal{C}[H_1] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H_3]$. □

Lemma E.3. *For any set $\mathbb{C}$ of contexts that is closed under plugging, and any preorder Q on natural numbers, the following holds.*

(1) *For any hypernets H_1 and H_2, $H_1 \simeq_{Q \cap Q^{-1}}^{\mathbb{C}} H_2$ implies $H_1 \simeq_Q^{\mathbb{C}} H_2$.*
(2) *If all compute transitions are deterministic, for any hypernets H_1 and H_2, $H_1 \simeq_Q^{\mathbb{C}} H_2$ implies $H_1 \simeq_{Q \cap Q^{-1}}^{\mathbb{C}} H_2$.*

Proof. Because $(Q \cap Q^{-1}) \subseteq Q$, the point (1) follows from the monotonicity of contextual equivalence.

For the point (2), $H_1 \simeq_Q^{\mathbb{C}} H_2$ means that any focus-free context $\mathcal{C}[\chi] \in \mathbb{C}$, such that $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_2]$ are states, yields $?;\mathcal{C}[H_1] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H_2]$ and $?;\mathcal{C}[H_2] \mathrel{\dot{\preceq}_Q} ?;\mathcal{C}[H_1]$. If the state $?;\mathcal{C}[H_1]$ terminates at a final state after k_1 transitions, there exists k_2 such that $k_1\ Q\ k_2$ and the state $?;\mathcal{C}[H_2]$ terminates at a final state after k_2 transitions. Moreover, there exists k_3 such that $k_2\ Q\ k_3$ and the state $?;\mathcal{C}[H_1]$ terminates at a final state after k_3 transitions.

Because search transitions and copy transitions are deterministic, if all compute transitions are deterministic, states and transitions comprise a deterministic abstract rewriting system, in which final states are normal forms. By Lemma C.1, $k_1 = k_3$ must hold. This means $k_1\ Q \cap Q^{-1}\ k_2$, and $?;\mathcal{C}[H_1] \mathrel{\dot{\preceq}_{Q \cap Q^{-1}}} ?;\mathcal{C}[H_2]$. Similarly, we can infer $?;\mathcal{C}[H_2] \mathrel{\dot{\preceq}_{Q \cap Q^{-1}}} ?;\mathcal{C}[H_1]$, and hence $H_1 \simeq_{Q \cap Q^{-1}}^{\mathbb{C}} H_2$. □

Appendix F. Proof of Theorem 8.11

This section details the coinductive proof of Theorem 8.11, with respect to the UAM $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ parameterised by $\mathbb{O}$ and $B_{\mathbb{O}}$.

At the core of the proof is step-wise reasoning, or transition-wise reasoning, using a lax variation of simulation dubbed *counting simulation*. Providing a simulation boils down to case analysis on transitions, namely on possible interactions between the focus and parts of states contributed by a pre-template. While output-closure helps us disprove some cases, input-safety and robustness give the cases that are specific to a pre-template and an operation set.

We also employ the so-called *up-to* technique in the use of quasi-specimens. We use counting simulations up to state refinements, with a quantitative restriction implemented by the notion of reasonable triple. This restriction is essential to make this particular up-to technique work, in combination with counting simulations. A similar form of up-to technique is studied categorically by Bonchi et al. [BPPR17], but for the ordinary notion of (weak) simulation, without this quantitative restriction.

The counting simulation up-to we use is namely (Q, Q', Q'')-*simulation*, parameterised by a triple (Q, Q', Q''). This provides a sound approach to prove state refinement $\dot{\preceq}_Q$, using

$\dot{\preceq}_{Q'}$ and $\dot{\preceq}_{Q''}$, given that all transitions are deterministic and (Q, Q', Q'') forms a reasonable triple.

Definition F.1 ((Q, Q', Q'')-simulations)**.** Let R be a binary relation on states, and (Q, Q', Q'') be a triple of preorders on $\mathbb{N}$. The binary relation R is a *Q-counting simulation up-to (Q', Q'')* (*(Q, Q', Q'')-simulation* in short) if, for any two related states $\dot{G}_1\ R\ \dot{G}_2$, the following (A) and (B) hold:

(A) If $\dot{G}_1$ is final, $\dot{G}_2$ is also final.

(B) If there exists a state $\dot{G}'_1$ such that $\dot{G}_1 \to \dot{G}'_1$, one of the following (I) and (II) holds:

(I) There exists a stuck state $\dot{G}''_1$ such that $\dot{G}'_1 \to^* \dot{G}''_1$.

(II) There exist two states $\dot{H}_1$ and $\dot{H}_2$, and numbers $k_1, k_2 \in \mathbb{N}$, such that $\dot{H}_1\ (\dot{\preceq}_{Q'} \circ R \circ \dot{\preceq}_{Q''})\ \dot{H}_2$, $(1 + k_1)\ Q\ k_2$, $\dot{G}'_1 \to^{k_1} \dot{H}_1$, and $\dot{G}_2 \to^{k_2} \dot{H}_2$.

Proposition F.2. *When the universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is deterministic, it satisfies the following.*

For any binary relation R on states, and any reasonable triple (Q, Q', Q''), if R is a (Q, Q', Q'')-simulation, then R implies refinement up to Q, i.e. any $\dot{G}_1\ R\ \dot{G}_2$ implies $\dot{G}_1 \dot{\preceq}_Q \dot{G}_2$.

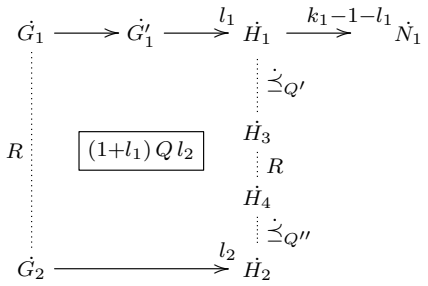
Proof. Our goal is to show the following: for any states $\dot{G}_1\ R\ \dot{G}_2$, any number $k_1 \in \mathbb{N}$ and any final state $\dot{N}_1$, such that $\dot{G}_1 \to^{k_1} \dot{N}_1$, there exist a number $k_2 \in \mathbb{N}$ and a final state $\dot{N}_2$ such that $k_1\ Q\ k_2$ and $\dot{G}_2 \to^{k_2} \dot{N}_2$. The proof is by induction on $k_1 \in \mathbb{N}$.

In the base case, when $k_1 = 0$, the state $\dot{G}_1$ is itself final because $\dot{G}_1 = \dot{N}_1$. Because R is a (Q, Q', Q'')-simulation, $\dot{G}_2$ is also a final state, which means we can take 0 as k_2 and $\dot{G}_2$ itself as $\dot{N}_2$. Because (Q, Q', Q'') is a reasonable triple, Q is a preorder and $0\ Q\ 0$ holds.

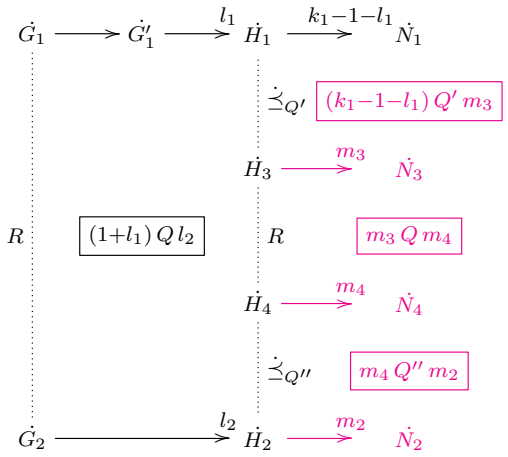
In the inductive case, when $k_1 > 0$, we assume the induction hypothesis on any $h \in \mathbb{N}$ such that $h < k_1$. Now that $k_1 > 0$, there exists a state $\dot{G}'_1$ such that $\dot{G}_1 \to \dot{G}'_1 \to^{k-1} \dot{N}_1$. Because all intrinsic transitions are deterministic, the assumption that compute transitions are all deterministic implies that states and transitions comprise a deterministic abstract rewriting system, in which final states and stuck states are normal forms. By Lemma C.1, we can conclude that there exists no stuck state $\dot{G}''_1$ such that $\dot{G}'_1 \to^* \dot{G}''_1$.

Therefore, by R being a (Q, Q', Q'')-simulation, there exist two states $\dot{H}_1$ and $\dot{H}_2$, and numbers $l_1, l_2 \in \mathbb{N}$, such that $\dot{H}_1\ (\dot{\preceq}_{Q'} \circ R \circ \dot{\preceq}_{Q''})\ \dot{H}_2$, $(1 + l_1)\ Q\ l_2$, $\dot{G}'_1 \to^{l_1} \dot{H}_1$, and $\dot{G}_2 \to^{l_2} \dot{H}_2$. By the determinism, $1 + l_1 \leq k_1$ must hold; if $\dot{H}_1$ is a final state, $\dot{G}'_1 \to^{l_1} \dot{H}_1$ must coincide with $\dot{G}'_1 \to^{k-1} \dot{N}_1$; otherwise, $\dot{G}'_1 \to^{l_1} \dot{H}_1$ must be a suffix of $\dot{G}'_1 \to^{k-1} \dot{N}_1$. There exist two states $\dot{H}_3$ and $\dot{H}_4$, and we have the following situation, where the relations

R, $\dot{\preceq}_{Q'}$ and $\dot{\preceq}_{Q''}$ are represented by vertical dotted lines from top to bottom.



We expand the above diagram as below (indicated by magenta), in three steps.



Firstly, by definition of state refinement, there exist a number $m_3 \in \mathbb{N}$ and a final state $\dot{N}_3$ such that $(k_1 - 1 - l_1)\ Q'\ m_3$ and $\dot{H}_3 \to^{m_3} \dot{N}_3$. Because (Q, Q', Q'') is a reasonable triple, $Q' \subseteq \geq_{\mathbb{N}}$, and hence $k_1 > k_1 - 1 - l_1 \geq m_3$. Therefore, secondly, by induction hypothesis on m_3, there exist a number $m_4 \in \mathbb{N}$ and a final state $\dot{N}_4$ such that $m_3\ Q\ m_4$ and $\dot{H}_4 \to^{m_4} \dot{N}_4$. Thirdly, by definition of state refinement, there exist a number $m_2 \in \mathbb{N}$ and a final state $\dot{N}_2$ such that $m_4\ Q''\ m_2$ and $\dot{H}_2 \to^{m_2} \dot{N}_2$.

Now we have $(k_1 - 1 - l_1)\ Q'\ m_3$, $m_3\ Q\ m_4$ and $m_4\ Q''\ m_2$, which means $(k_1 - 1 - l_1)\ (Q' \circ Q \circ Q'')\ m_2$. Because (Q, Q', Q'') is a reasonable triple, this implies $(k_1{-}1{-}l_1)\ Q\ m_2$, and moreover, $k_1\ Q\ (l_2 + m_2)$. We can take $l_2 + m_2$ as k_2. □

The focus in a focussed context $\dot{\mathcal{C}}$ is said to be *remote*, if it is not the entering $\lightning$-focus. The procedure of *contextual lifting* reduces a proof of contextual refinement down to that of state refinement.

Definition F.3 (Contextual lifting)**.** Let $\mathbb{C} \subseteq \mathcal{H}_\omega(L, M_{\mathbb{O}} \cup \mathbb{M})$ be a set of contexts. Given a pre-template $\lhd$ on focus-free hypernets $\mathcal{H}_\omega(L, M_{\mathbb{O}} \backslash \{?, \checkmark, \lightning\})$, its $\mathbb{C}$-*contextual lifting* $\overline{\lhd}^{\mathbb{C}}$ is a binary relation on states defined by: $\dot{G}_1\ \overline{\lhd}^{\mathbb{C}}\ \dot{G}_2$ if there exists a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$ of $\lhd$, such that the focus of $\dot{\mathcal{C}}$ is remote, $\dot{G}_p = \dot{\mathcal{C}}[\vec{H}^p]$, and $\dot{\mathcal{C}}[\vec{H}^p]$ is rooted, for each $p \in \{1, 2\}$.

The contextual lifting $\overline{\lhd}^{\mathbb{C}}$ is by definition a binary relation on rooted states.

Proposition F.4. *For any set $\mathbb{C} \subseteq \mathcal{H}_\omega(L, M_\mathbb{O} \cup \mathbb{M})$ of contexts that is closed under plugging, any preorder Q on $\mathbb{N}$, and any pre-template $\lhd$ on focus-free hypernets $\mathcal{H}_\omega(L, M_\mathbb{O} \backslash \{?, \checkmark, \lightning\})$, if the $\mathbb{C}$-contextual lifting $\overline{\lhd}^{\mathbb{C}}$ implies refinement $\dot{\preceq}_Q$ (resp. equivalence $\dot{\simeq}_Q$), then $\lhd$ implies contextual refinement $\preceq_Q^{\mathbb{C}}$ (resp. contextual equivalence $\simeq_Q^{\mathbb{C}}$).*

Proof of refinement case. Our goal is to show that, for any $H_1 \lhd H_2$ and any focus-free context $\mathcal{C}[\chi] \in \mathbb{C}$ such that $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_2]$ are states, we have refinement $?;\mathcal{C}[H_1] \dot{\preceq}_Q ?;\mathcal{C}[H_2]$.

Because $?;\mathcal{C}[H_p] = ?;(\mathcal{C}[H_p]) = (?;\mathcal{C})[H_p]$ for $p \in \{1, 2\}$, and $|?;\mathcal{C}| = \mathcal{C} \in \mathbb{C}$, the triple $((?;\mathcal{C}); H_1; H_2)$ is a $\mathbb{C}$-specimen of $\lhd$ with the ?-focus. Moreover the states $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_2]$ are trivially rooted. Therefore, $?;\mathcal{C}[H_1] \overline{\lhd}^{\mathbb{C}} ?;\mathcal{C}[H_2]$, and by the assumption, $?;\mathcal{C}[H_1] \dot{\preceq}_Q ?;\mathcal{C}[H_2]$. □

Proof of equivalence case. It suffices to show that, for any $H_1 \lhd H_2$ and any focus-free context $\mathcal{C}[\chi] \in \mathbb{C}$ such that $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_2]$ are states, we have refinements $?;\mathcal{C}[H_1] \dot{\preceq}_Q ?;\mathcal{C}[H_2]$ and $?;\mathcal{C}[H_2] \dot{\preceq}_Q ?;\mathcal{C}[H_1]$, i.e. equivalence $?;\mathcal{C}[H_1] \dot{\simeq}_Q ?;\mathcal{C}[H_2]$.

Because $?;\mathcal{C}[H_p] = ?;(\mathcal{C}[H_p]) = (?;\mathcal{C})[H_p]$ for $p \in \{1, 2\}$, and $|?;\mathcal{C}| = \mathcal{C} \in \mathbb{C}$, the triple $((?;\mathcal{C}); H_1; H_2)$ is a $\mathbb{C}$-specimen of $\lhd$ with the ?-focus. Moreover the states $?;\mathcal{C}[H_1]$ and $?;\mathcal{C}[H_2]$ are trivially rooted. Therefore, $?;\mathcal{C}[H_1] \overline{\lhd}^{\mathbb{C}} ?;\mathcal{C}[H_2]$, and by the assumption, $?;\mathcal{C}[H_1] \dot{\simeq}_Q ?;\mathcal{C}[H_2]$. □

Lemma F.5. *For any set $\mathbb{C} \subseteq \mathcal{H}_\omega(L, M_\mathbb{O} \cup \mathbb{M})$ of contexts that is closed under plugging, any pre-template $\lhd$ on focus-free hypernets $\mathcal{H}_\omega(L, M_\mathbb{O} \backslash \{?, \checkmark, \lightning\})$, and any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\vec{\chi}]; \vec{H^1}; \vec{H^2})$ of $\lhd$, the following holds.*

1. *The state $\dot{\mathcal{C}}[\vec{H^1}]$ is final (resp. initial) if and only if the state $\dot{\mathcal{C}}[\vec{H^2}]$ is final (resp. initial).*
2. *If $\lhd$ is output-closed, and $\dot{\mathcal{C}}[\vec{H^1}]$ and $\dot{\mathcal{C}}[\vec{H^2}]$ are both rooted states, then the focus of $\dot{\mathcal{C}}$ is not exiting.*
3. *If $\lhd$ is output-closed, $\dot{\mathcal{C}}[\vec{H^1}]$ and $\dot{\mathcal{C}}[\vec{H^2}]$ are both rooted states, the focus of $\dot{\mathcal{C}}$ is the $\checkmark$-focus or the non-entering ?-focus, and a transition is possible from $\dot{\mathcal{C}}[\vec{H^1}]$ or $\dot{\mathcal{C}}[\vec{H^2}]$, then there exists a focussed context $\dot{\mathcal{C}}'$ with a remote focus such that $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}|$ and $\dot{\mathcal{C}}[\vec{H^p}] \to \dot{\mathcal{C}}'[\vec{H^p}]$ for each $p \in \{1, 2\}$.*

Proof of point (1). Let (p, q) be an arbitrary element of a set $\{(1, 2), (2, 1)\}$. If $\dot{\mathcal{C}}[\vec{H^p}]$ is final (resp. initial), the focus source is an input in $\dot{\mathcal{C}}[\vec{H^p}]$. Because input lists of $\dot{\mathcal{C}}[\vec{H^p}]$, $\dot{\mathcal{C}}$ and $\dot{\mathcal{C}}[\vec{H^q}]$ all coincide, the focus source must be an input in $\dot{\mathcal{C}}$, and in $\dot{\mathcal{C}}[\vec{H^q}]$ too. This means $\dot{\mathcal{C}}[\vec{H^q}]$ is also a final (resp. initial) state. □

Proof of point (2). This is a consequence of the contraposition of Lemma C.7(3). □

Proof of the point (3). The transition possible from $\dot{\mathcal{C}}[\vec{H^1}]$ or $\dot{\mathcal{C}}[\vec{H^2}]$ is necessarily a search transition. By case analysis on the focus of $\dot{\mathcal{C}}$, we can confirm that the search transition applies an interaction rule to the focus and an edge from $\dot{\mathcal{C}}$.

- When the focus of $\dot{\mathcal{C}}$ is the $\checkmark$-focus, the transition can only change the focus and its incoming operation edge. Because $\lhd$ is output-closed, by the point (2), the focus of $\dot{\mathcal{C}}$ is not exiting. This implies that the incoming operation edge of the focus is from $\dot{\mathcal{C}}$ in both states $\dot{\mathcal{C}}[\vec{H^1}]$ and $\dot{\mathcal{C}}[\vec{H^2}]$.

- When the focus of $\dot{\mathcal{C}}$ is the non-entering ?-focus, the transition can only change the focus and its outgoing edge. Because the focus is not entering in $\dot{\mathcal{C}}$, the outgoing edge is from $\dot{\mathcal{C}}$ in both states $\dot{\mathcal{C}}[\vec{H}^1]$ and $\dot{\mathcal{C}}[\vec{H}^2]$.

Therefore, there exist a focus-free simple context $\mathcal{C}_0[\chi, \vec{\chi}]$ and an interaction rule $\dot{N}_0 \mapsto \dot{N}_0'$, such that $\dot{\mathcal{C}} = \mathcal{C}_0[\dot{N}_0, \vec{\chi}]$, and $\mathcal{C}_0[\dot{N}_0', \vec{\chi}]$ is a focussed context.

Examining interaction rules confirms $|\dot{N}_0| = |\dot{N}_0'|$, and hence $|\dot{\mathcal{C}}| = |\mathcal{C}_0[\dot{N}_0, \vec{\chi}]| = |\mathcal{C}_0[\dot{N}_0', \vec{\chi}]|$. By definition of search transitions, we have:

$$\dot{\mathcal{C}}[\vec{H}^p] = \mathcal{C}_0[\dot{N}_0, \vec{H}^p] \to \mathcal{C}_0[\dot{N}_0', \vec{H}^p]$$

for each $p \in \{1, 2\}$.

The rest of the proof is to check that $\mathcal{C}_0[\dot{N}_0', \vec{\chi}]$ has a remote focus, namely that, if its focus is the $\lightning$-focus, the focus is not entering. This is done by inspecting interaction rules.

- When the interaction rule $\dot{N}_0 \mapsto \dot{N}_0'$ changes the ✓-focus to the $\lightning$-focus, this must be the interaction rule in Figure 9d, which means $\dot{N}_0'$ consists of the $\lightning$-focus and its outgoing operation edge. The operation edge remains to be a (unique) outgoing edge of the focus in $\mathcal{C}_0[\dot{N}_0', \vec{\chi}]$, and hence the focus is not entering in $\mathcal{C}_0[\dot{N}_0', \vec{\chi}]$.
- When the interaction rule $\dot{N}_0 \mapsto \dot{N}_0'$ changes the ?-focus to the $\lightning$-focus, this must be the interaction rule in Figure 18a, 18b, 9e, which means $\dot{N}_0' = \langle \dot{N}_0 \rangle_{\lightning/?}$. Because the focus is not entering in $\mathcal{C}_0[\dot{N}_0, \vec{\chi}] = \dot{\mathcal{C}}$, the focus is also not entering in $\mathcal{C}_0[\dot{N}_0', \vec{\chi}] = \langle \mathcal{C}_0[\dot{N}_0, \vec{\chi}] \rangle_{\lightning/?}$. □

Proposition F.6. *When the universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ is deterministic and refocusing, it satisfies the following, for any set $\mathbb{C} \subseteq \mathcal{H}_\omega(L, M_{\mathbb{O}} \cup \mathbb{M})$ of contexts that is closed under plugging, any reasonable triple (Q, Q', Q''), and any pre-template $\lhd$ on focus-free hypernets $\mathcal{H}_\omega(L, M_{\mathbb{O}} \backslash \{?, ✓, \lightning\})$.*

(1) *If $\lhd$ is a $(\mathbb{C}, Q, Q')$-template and $(\mathbb{C}, Q, Q', Q'')$-robust relative to all rewrite transitions, then the $\mathbb{C}$-contextual lifting $\overline{\lhd}^{\mathbb{C}}$ is a (Q, Q', Q'')-simulation.*
(2) *If $\lhd$ is a $(\mathbb{C}, Q^{-1}, Q')$-template and the converse $\lhd^{-1}$ is $(\mathbb{C}, Q, Q', Q'')$-robust relative to all rewrite transitions, then the $\mathbb{C}$-contextual lifting $\overline{\lhd^{-1}}^{\mathbb{C}}$ of the converse is a (Q, Q', Q'')-simulation.*

Proof prelude. Let $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$ be an arbitrary $\mathbb{C}$-specimen of $\lhd$, such that the focus of $\dot{\mathcal{C}}$ is remote, and $\dot{G}_p := \dot{\mathcal{C}}[\vec{H}^p]$ is a rooted state for each $p \in \{1, 2\}$. By definition of contextual lifting, $\dot{G}_1 \ \overline{\lhd}^{\mathbb{C}} \ \dot{G}_2$, and equivalently, $\dot{G}_2 \ (\overline{\lhd}^{\mathbb{C}})^{-1} \ \dot{G}_1$. Note that $\overline{\lhd^{-1}}^{\mathbb{C}} = (\overline{\lhd}^{\mathbb{C}})^{-1}$.

Because $\lhd$ is output-closed, by Lemma F.5(2), the focus is not exiting in $\dot{\mathcal{C}}$. This implies that, if the focus has an incoming edge in $\dot{G}_1$ or $\dot{G}_2$, the incoming edge must be from $\dot{\mathcal{C}}$.

Because the machine is deterministic and refocusing, rooted states and transitions comprise a deterministic abstract rewriting system, in which final states and stuck states are normal forms. By Lemma C.1, from any state, a sequence of transitions that result in a final state or a stuck state is unique, if any.

Because (Q, Q', Q'') is a reasonable triple, Q' and Q'' are reflexive. By Lemma E.2, this implies that $\dot{\preceq}_{Q'}$ and $\dot{\preceq}_{Q''}$ are reflexive, and hence $\overline{\lhd}^{\mathbb{C}} \subseteq \dot{\preceq}_{Q'} \circ \overline{\lhd}^{\mathbb{C}} \circ \dot{\preceq}_{Q''}$, and $(\overline{\lhd}^{\mathbb{C}})^{-1} \subseteq \dot{\preceq}_{Q'} \circ (\overline{\lhd}^{\mathbb{C}})^{-1} \circ \dot{\preceq}_{Q''}$. □

Proof of the point (1). Our goal is to check conditions (A) and (B) of Definition F.1 for the states $\dot{G}_1 \ \overline{\lhd}^{\mathbb{C}} \ \dot{G}_2$.

If $\dot{G}_1$ is final, by Lemma F.5(1), $\dot{G}_2$ is also final. The condition (A) of Definition F.1 is fulfilled.

If there exists a state $\dot{G}'_1$ such that $\dot{G}_1 \to \dot{G}'_1$, we show that one of the conditions (I) and (II) of Definition F.1 is fulfilled, by case analysis of the focus in $\dot{\mathcal{C}}$.

- When the focus is the ✓-focus, or the ?-focus that is not entering, by Lemma F.5(3), there exists a focussed context $\dot{\mathcal{C}}'$ with a remote focus, such that $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}|$ and $\dot{G}_p = \dot{\mathcal{C}}[\vec{H}^p] \to \dot{\mathcal{C}}'[\vec{H}^p]$ for each $p \in \{1, 2\}$. We have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II) of Definition F.1 is fulfilled.

$$\begin{array}{ccc} \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & \longrightarrow & \dot{\mathcal{C}}'[\vec{H}^1] = \dot{G}'_1 \\ \overline{\lhd}^{\mathbb{C}} \vdots & \boxed{1\ Q\ 1} & \vdots \overline{\lhd}^{\mathbb{C}} \\ \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^2] & \longrightarrow & \dot{\mathcal{C}}'[\vec{H}^2] \end{array}$$

  By the determinism, $\dot{\mathcal{C}}'[\vec{H}^1] = \dot{G}'_1$. Because (Q, Q', Q'') is a reasonable triple, Q is a preorder and $1\ Q\ 1$. The context $\dot{\mathcal{C}}'$ satisfies $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}| \in \mathbb{C}$, so $(\dot{\mathcal{C}}'; \vec{H}^1; \vec{H}^2)$ is a $\mathbb{C}$-specimen of $\lhd$. The context $\dot{\mathcal{C}}'$ has a remote focus, and the states $\dot{\mathcal{C}}'[\vec{H}^1]$ and $\dot{\mathcal{C}}'[\vec{H}^2]$ are both rooted. Therefore, we have $\dot{\mathcal{C}}'[\vec{H}^1] \overline{\lhd}^{\mathbb{C}} \dot{\mathcal{C}}'[\vec{H}^2]$.
- When the focus is the ?-focus that is entering in $\dot{\mathcal{C}}$, because $\lhd$ is $(\mathbb{C}, Q, Q')$-input-safe, we have one of the following three situations corresponding to (I), (II) and (III) of Definition 8.4.
  - There exist two stuck states $\dot{N}_1$ and $\dot{N}_2$ such that $\dot{G}_p \to^* \dot{N}_p$ for each $p \in \{1, 2\}$. By the determinism of transitions, we have $\dot{G}_1 \to \dot{G}'_1 \to^* \dot{N}_1$, which means the condition (I) of Definition F.1 is satisfied.
  - There exist a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'}^1; \vec{H'}^2)$ of $\lhd$ and two numbers $k_1, k_2 \in \mathbb{N}$, such that the focus of $\dot{\mathcal{C}}'$ is the ✓-focus or the non-entering ?-focus, $(1{+}k_1)\ Q\ k_2$, $\dot{\mathcal{C}}[\vec{H}^1] \to^{1+k_1} \dot{\mathcal{C}}'[\vec{H'}^1]$, and $\dot{\mathcal{C}}[\vec{H}^2] \to^{k_2} \dot{\mathcal{C}}'[\vec{H'}^2]$. By the determinism of transitions, we have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II) of Definition F.1 is fulfilled.

$$\begin{array}{ccccc} \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & \longrightarrow & \dot{G}'_1 & \xrightarrow{k_1} & \dot{\mathcal{C}}'[\vec{H'}^1] \\ \overline{\lhd}^{\mathbb{C}} \vdots & & \boxed{(1{+}k_1)\ Q\ k_2} & & \vdots \overline{\lhd}^{\mathbb{C}} \\ \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^2] & & \xrightarrow{k_2} & & \dot{\mathcal{C}}'[\vec{H'}^2] \end{array}$$

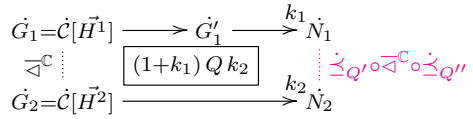
    The context $\dot{\mathcal{C}}'$ has a remote focus, and states $\dot{\mathcal{C}}'[\vec{H'}^1]$ and $\dot{\mathcal{C}}'[\vec{H'}^2]$ are rooted. Therefore, $\dot{\mathcal{C}}'[\vec{H'}^1] \overline{\lhd}^{\mathbb{C}} \dot{\mathcal{C}}'[\vec{H'}^2]$.
  - There exist a quasi-$\mathbb{C}$-specimen $(\dot{N}_1, \dot{N}_2)$ of $\lhd$ up to $(\dot{\simeq}_{Q'}, \dot{\simeq}_{Q'})$, whose focus is not the $\lightning$-focus, and two numbers $k_1, k_2 \in \mathbb{N}$, such that $(1 + k_1)\ Q\ (1 + k_2)$, $\dot{\mathcal{C}}[\vec{H}^1] \to^{1+k_1} \dot{N}_1$, and $\dot{\mathcal{C}}[\vec{H}^2] \to^{1+k_2} \dot{N}_2$. By the determinism of transitions, we have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II) of Definition F.1 is fulfilled.

$$\begin{array}{ccccc} \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & \longrightarrow & \dot{G}'_1 & \xrightarrow{k_1} & \dot{N}_1 \\ \overline{\lhd}^{\mathbb{C}} \vdots & & \boxed{(1{+}k_1)\ Q\ (1{+}k_2)} & & \vdots\ \dot{\simeq}_{Q'} \circ \overline{\lhd}^{\mathbb{C}} \circ \dot{\simeq}_{Q''} \\ \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^2] & & \xrightarrow{1+k_2} & & \dot{N}_2 \end{array}$$

Because $(\dot{N}_1, \dot{N}_2)$ is a quasi-$\mathbb{C}$-specimen of $\lhd$ up to $(\doteq_{Q'}, \doteq_{Q'})$, and states $\dot{N}_1$ and $\dot{N}_2$ are rooted, there exists a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'^1}; \vec{H'^2})$ of $\lhd$ with a non-$\lightning$ focus, such that $\dot{\mathcal{C}}'[\vec{H'^1}]$ and $\dot{\mathcal{C}}'[\vec{H'^2}]$ are also rooted, $\dot{N}_1 \doteq_{Q'} \dot{\mathcal{C}}'[\vec{H'^1}]$, and $\dot{\mathcal{C}}'[\vec{H'^2}] \doteq_{Q'} \dot{N}_2$. Because (Q, Q', Q'') is a reasonable triple, $Q' \subseteq Q''$, and hence $\doteq_{Q'} \subseteq \doteq_{Q''}$. Therefore, we have:

$$\dot{N}_1 \doteq_{Q'} \dot{\mathcal{C}}'[\vec{H'^1}] \mathrel{\overline{\lhd}^{\mathbb{C}}} \dot{\mathcal{C}}'[\vec{H'^2}] \doteq_{Q''} \dot{N}_2.$$

- When the focus is the $\lightning$-focus, $\dot{G}_1 \to \dot{G}'_1$ is a rewrite transition, and by definition of contextual lifting, the focus is not entering in $\dot{\mathcal{C}}$. Because $\lhd$ is $(\mathbb{C}, Q, Q', Q'')$-robust relative to all rewrite transitions, and $\dot{G}_1$ and $\dot{G}_2$ are rooted, we have one of the following two situations corresponding to (II) and (III) of Definition 8.8.
  - There exists a stuck state $\dot{N}$ such that $\dot{G}'_1 \to^* \dot{N}$. The condition (I) of Definition F.1 is satisfied.
  - There exist a quasi-$\mathbb{C}$-specimen $(\dot{N}_1, \dot{N}_2)$ of $\lhd$ up to $(\dot{\preceq}_{Q'}, \dot{\preceq}_{Q''})$, whose focus is not the $\lightning$-focus, and two numbers $k_1, k_2 \in \mathbb{N}$, such that $(1 + k_1)\ Q\ k_2$, $\dot{G}'_1 \to^{k_1} \dot{N}_1$, and $\dot{G}_2 \to^{k_2} \dot{N}_2$. We have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II) of Definition F.1 is fulfilled.



  Because $(\dot{N}_1, \dot{N}_2)$ is a quasi-$\mathbb{C}$-specimen of $\lhd$ up to $(\dot{\preceq}_{Q'}, \dot{\preceq}_{Q''})$, and states $\dot{N}_1$ and $\dot{N}_2$ are rooted, there exists a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'^1}; \vec{H'^2})$ of $\lhd$ with a non-$\lightning$ focus, such that $\dot{\mathcal{C}}'[\vec{H'^1}]$ and $\dot{\mathcal{C}}'[\vec{H'^2}]$ are also rooted, $\dot{N}_1 \doteq_{Q'} \dot{\mathcal{C}}'[\vec{H'^1}]$, and $\dot{\mathcal{C}}'[\vec{H'^2}] \doteq_{Q''} \dot{N}_2$. This means $\dot{\mathcal{C}}'[\vec{H'^1}] \mathrel{\overline{\lhd}^{\mathbb{C}}} \dot{\mathcal{C}}'[\vec{H'^2}]$, and hence:

$$\dot{N}_1 \doteq_{Q'} \dot{\mathcal{C}}'[\vec{H'^1}] \mathrel{\overline{\lhd}^{\mathbb{C}}} \dot{\mathcal{C}}'[\vec{H'^2}] \doteq_{Q''} \dot{N}_2. \qquad \square$$

Proof of the point (2). It suffices to check the "reverse" of conditions (A) and (B) of Definition F.1 for the states $\dot{G}_2\ (\overline{\lhd}^{\mathbb{C}})^{-1}\ \dot{G}_1$, namely the following conditions (A') and (B').

(A') If $\dot{G}_2$ is final, $\dot{G}_1$ is also final.

(B') If there exists a state $\dot{G}'_2$ such that $\dot{G}_2 \to \dot{G}'_2$, one of the following (I') and (II') holds.

(I') There exists a stuck state $\dot{G}''_2$ such that $\dot{G}'_2 \to^* \dot{G}''_2$.

(II') There exist two states $\dot{N}_2$ and $\dot{N}_1$, and numbers $k_2, k_1 \in \mathbb{N}$, such that $\dot{N}_2\ (\dot{\preceq}_{Q'} \circ (\overline{\lhd}^{\mathbb{C}})^{-1} \circ \dot{\preceq}_{Q'})\ \dot{N}_1$, $(1 + k_2)\ Q\ k_1$, $\dot{G}'_2 \to^{k_2} \dot{N}_2$, and $\dot{G}_1 \to^{k_1} \dot{N}_1$.

The proof is mostly symmetric to the point (1). Note that there is a one-to-one correspondence between $\mathbb{C}$-specimens of $\lhd$ and $\mathbb{C}$-specimens of $\lhd^{-1}$; any $\mathbb{C}$-specimen $(\dot{\mathcal{C}}_0; \vec{H^{01}}; \vec{H^{02}})$ of $\lhd$ gives a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}_0; \vec{H^{02}}; \vec{H^{01}})$ of $\lhd^{-1}$. Because $\lhd$ is output-closed, its converse $\lhd^{-1}$ is also output-closed.

If $\dot{G}_2$ is final, by Lemma F.5(1), $\dot{G}_1$ is also final. The condition (A') is fulfilled.

If there exists a state $\dot{G}'_2$ such that $\dot{G}_2 \to \dot{G}'_2$, we show that one of the conditions (I') and (II') above is fulfilled, by case analysis of the focus in $\dot{\mathcal{C}}$.

- When the focus is the ✓-focus, or the ?-focus that is not entering, by Lemma F.5(3), there exists a focussed context $\dot{\mathcal{C}}'$ with a remote focus, such that $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}|$ and $\dot{G}_p = \dot{\mathcal{C}}[\vec{H^p}] \to$

$\dot{\mathcal{C}}'[\vec{H}^p]$ for each $p \in \{1,2\}$. We have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II') is fulfilled.

$$\begin{array}{ccc} \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^1] & \longrightarrow & \dot{\mathcal{C}}'[\vec{H}^2] = \dot{G}'_2 \\ (\overline{\lhd}^{\mathbb{C}})^{-1} \vdots & \boxed{1\,Q\,1} & \vdots (\overline{\lhd}^{\mathbb{C}})^{-1} \\ \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & \longrightarrow & \dot{\mathcal{C}}'[\vec{H}^1] \end{array}$$

By the determinism, $\dot{\mathcal{C}}'[\vec{H}^2] = \dot{G}'_2$. Because (Q, Q', Q'') is a reasonable triple, Q is a preorder and $1\ Q\ 1$. The context $\dot{\mathcal{C}}'$ satisfies $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}| \in \mathbb{C}$, so $(\dot{\mathcal{C}}'; \vec{H}^2; \vec{H}^1)$ is a $\mathbb{C}$-specimen of $\lhd^{-1}$. The context $\dot{\mathcal{C}}'$ has a remote focus, and the states $\dot{\mathcal{C}}'[\vec{H}^1]$ and $\dot{\mathcal{C}}'[\vec{H}^2]$ are both rooted. Therefore, we have $\dot{\mathcal{C}}'[\vec{H}^2]\ (\overline{\lhd}^{\mathbb{C}})^{-1}\ \dot{\mathcal{C}}'[\vec{H}^1]$.

- When the focus is the ?-focus that is entering in $\dot{\mathcal{C}}$, because $\lhd$ is $(\mathbb{C}, Q^{-1}, Q')$-input-safe, we have one of the following three situations corresponding to (I), (II) and (III) of Definition 8.4.
  - There exist two stuck states $\dot{N}_1$ and $\dot{N}_2$ such that $\dot{G}_p \to^* \dot{N}_p$ for each $p \in \{1,2\}$. By the determinism of transitions, we have $\dot{G}_2 \to \dot{G}'_2 \to^* \dot{N}_2$, which means the condition (I') is satisfied.
  - There exist a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'}^1; \vec{H'}^2)$ of $\lhd$ and two numbers $k_1, k_2 \in \mathbb{N}$, such that the focus of $\dot{\mathcal{C}}'$ is the ✓-focus or the non-entering ?-focus, $(1+k_1)\ Q^{-1}\ k_2$, $\dot{\mathcal{C}}[\vec{H}^1] \to^{1+k_1} \dot{\mathcal{C}}'[\vec{H'}^1]$, and $\dot{\mathcal{C}}[\vec{H}^2] \to^{k_2} \dot{\mathcal{C}}'[\vec{H'}^2]$. We have the following situation, namely the black part of the diagram below.

$$\begin{array}{ccc} \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^2] & \xrightarrow{\quad k_2 \quad} & \dot{\mathcal{C}}'[\vec{H'}^2] \\ (\overline{\lhd}^{\mathbb{C}})^{-1} \vdots & \boxed{k_2\,Q\,(1{+}k_1)} & \vdots (\overline{\lhd}^{\mathbb{C}})^{-1} \\ \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & \xrightarrow{\quad 1+k_1 \quad} & \dot{\mathcal{C}}'[\vec{H'}^1] \end{array}$$

The magenta part holds, because the focus of $\dot{\mathcal{C}}'$ is not the $\lightning$-focus and not entering, and because states $\dot{\mathcal{C}}'[\vec{H'}^1]$ and $\dot{\mathcal{C}}'[\vec{H'}^2]$ are rooted. We check the condition (II') by case analysis on the number k_2.

  - When $k_2 > 0$, by the determinism of transitions, we have the following diagram, which means the condition (II') is fulfilled.

$$\begin{array}{ccccc} \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^2] & \longrightarrow & \dot{G}'_2 & \xrightarrow{\quad k_2-1 \quad} & \dot{\mathcal{C}}'[\vec{H'}^2] \\ (\overline{\lhd}^{\mathbb{C}})^{-1} \vdots & & \boxed{k_2\,Q\,(1{+}k_1)} & & \vdots (\overline{\lhd}^{\mathbb{C}})^{-1} \\ \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & & \xrightarrow{\quad 1+k_1 \quad} & & \dot{\mathcal{C}}'[\vec{H'}^1] \end{array}$$

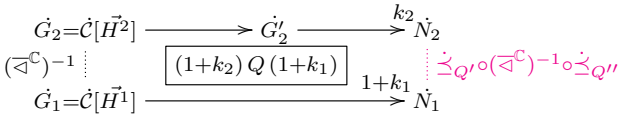
  - When $k_2 = 0$, $\dot{G}_2 = \dot{\mathcal{C}}[\vec{H}^2] = \dot{\mathcal{C}}'[\vec{H'}^2]$, and we have the following situation, namely the black part of the diagram below.

$$\begin{array}{ccccc} \dot{G}_2{=}\dot{\mathcal{C}}[\vec{H}^2] & \xrightarrow{\quad 0 \quad} & \dot{G}_2{=}\dot{\mathcal{C}}'[\vec{H'}^2] & \longrightarrow & \dot{G}'_2{=}\dot{\mathcal{C}}''[\vec{H'}^1] \\ (\overline{\lhd}^{\mathbb{C}})^{-1} \vdots & \boxed{0\,Q\,(1{+}k_1)} & (\overline{\lhd}^{\mathbb{C}})^{-1}\boxed{1\,Q\,1} & & \vdots (\overline{\lhd}^{\mathbb{C}})^{-1} \\ \dot{G}_1{=}\dot{\mathcal{C}}[\vec{H}^1] & \xrightarrow{\quad 1+k_1 \quad} & \dot{\mathcal{C}}'[\vec{H'}^1] & \longrightarrow & \dot{\mathcal{C}}''[\vec{H'}^1] \end{array}$$

Because $\dot{G}_2 \to \dot{G}'_2$, and the focus of $\dot{\mathcal{C}}'$ is the ✓-focus, or the non-entering ?-focus, by Lemma F.5(3), there exists a focussed context $\dot{\mathcal{C}}''$ with a remote focus, such

that $|\dot{\mathcal{C}}''| = |\dot{\mathcal{C}}'|$ and $\dot{\mathcal{C}}'[\vec{H'}^p] \to \dot{\mathcal{C}}''[\vec{H'}^p]$ for each $p \in \{1,2\}$. By the determinism of transitions, $\dot{G}'_2 = \dot{\mathcal{C}}''[\vec{H'}^1]$. Because (Q, Q', Q'') is a reasonable triple, Q is a preorder and $1\ Q\ 1$. The context $\dot{\mathcal{C}}''$ satisfies $|\dot{\mathcal{C}}''| = |\dot{\mathcal{C}}'| \in \mathbb{C}$, so $(\dot{\mathcal{C}}''; \vec{H'}^2; \vec{H'}^1)$ is a $\mathbb{C}$-specimen of $\lhd^{-1}$. The context $\dot{\mathcal{C}}''$ has a remote focus, and the states $\dot{\mathcal{C}}''[\vec{H'}^1]$ and $\dot{\mathcal{C}}''[\vec{H'}^2]$ are both rooted. Therefore, we have $\dot{\mathcal{C}}''[\vec{H'}^2]\ (\overline{\lhd}^{\mathbb{C}})^{-1}\ \dot{\mathcal{C}}''[\vec{H'}^1]$. Finally, because (Q, Q', Q'') is a reasonable triple, Q is closed under addition, and hence $1\ Q\ (2 + k_1)$. The condition (II') is fulfilled.

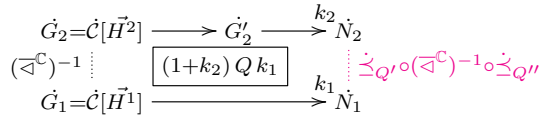
– There exist a quasi-$\mathbb{C}$-specimen $(\dot{N}_1, \dot{N}_2)$ of $\lhd$ up to $(\dot{\simeq}_{Q'}, \dot{\simeq}_{Q'})$, whose focus is not the $\lightning$-focus, and two numbers $k_1, k_2 \in \mathbb{N}$, such that $(1 + k_1)\ Q^{-1}\ (1 + k_2)$, $\dot{\mathcal{C}}[\vec{H}^1] \to^{1+k_1} \dot{N}_1$, and $\dot{\mathcal{C}}[\vec{H}^2] \to^{1+k_2} \dot{N}_2$. By the determinism of transitions, we have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II') is fulfilled.



Because $(\dot{N}_1, \dot{N}_2)$ is a quasi-$\mathbb{C}$-specimen of $\lhd$ up to $(\dot{\simeq}_{Q'}, \dot{\simeq}_{Q'})$, and states $\dot{N}_1$ and $\dot{N}_2$ are rooted, there exists a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'}^1; \vec{H'}^2)$ of $\lhd$ with a non-$\lightning$ focus, such that $\dot{\mathcal{C}}'[\vec{H'}^1]$ and $\dot{\mathcal{C}}'[\vec{H'}^2]$ are also rooted, $\dot{N}_1 \dot{\simeq}_{Q'} \dot{\mathcal{C}}'[\vec{H'}^1]$, and $\dot{\mathcal{C}}'[\vec{H'}^2] \dot{\simeq}_{Q'} \dot{N}_2$. Because (Q, Q', Q'') is a reasonable triple, $Q' \subseteq Q''$, and hence $\dot{\simeq}_{Q'} \subseteq \dot{\simeq}_{Q''}$. Therefore, we have:

$$\dot{N}_2 \dot{\preceq}_{Q'} \dot{\mathcal{C}}'[\vec{H'}^2]\ (\overline{\lhd}^{\mathbb{C}})^{-1}\ \dot{\mathcal{C}}'[\vec{H'}^1] \dot{\preceq}_{Q''} \dot{N}_1.$$

- When the focus is the $\lightning$-focus, $\dot{G}_2 \to \dot{G}'_2$ is a rewrite transition, and by definition of contextual lifting, the focus is not entering in $\dot{\mathcal{C}}$. Because $\lhd^{-1}$ is $(\mathbb{C}, Q, Q', Q'')$-robust relative to all rewrite transitions, and $\dot{G}_1$ and $\dot{G}_2$ are rooted, we have one of the following two situations corresponding to (II) and (III) of Definition 8.8.
  – There exists a stuck state $\dot{N}$ such that $\dot{G}'_2 \to^* \dot{N}$. The condition (I') is satisfied.
  – There exist a quasi-$\mathbb{C}$-specimen $(\dot{N}_2, \dot{N}_1)$ of $\lhd^{-1}$ up to $(\dot{\preceq}_{Q'}, \dot{\preceq}_{Q''})$, whose focus is not the $\lightning$-focus, and two numbers $k_2, k_1 \in \mathbb{N}$, such that $(1 + k_2)\ Q\ k_1$, $\dot{G}'_2 \to^{k_2} \dot{N}_2$, and $\dot{G}_1 \to^{k_1} \dot{N}_1$. We have the following situation, namely the black part of the diagram below. Showing the magenta part confirms that the condition (II') is fulfilled.



Because $(\dot{N}_2, \dot{N}_1)$ is a quasi-$\mathbb{C}$-specimen of $\lhd^{-1}$ up to $(\dot{\preceq}_{Q'}, \dot{\preceq}_{Q''})$, and states $\dot{N}_2$ and $\dot{N}_1$ are rooted, there exists a $\mathbb{C}$-specimen $(\dot{\mathcal{C}}'; \vec{H'}^2; \vec{H'}^1)$ of $\lhd^{-1}$ with a non-$\lightning$ focus, such that $\dot{\mathcal{C}}'[\vec{H'}^2]$ and $\dot{\mathcal{C}}'[\vec{H'}^1]$ are also rooted, $\dot{N}_2 \dot{\preceq}_{Q'} \dot{\mathcal{C}}'[\vec{H'}^2]$, and $\dot{\mathcal{C}}'[\vec{H'}^1] \dot{\preceq}_{Q''} \dot{N}_1$. This means $\dot{\mathcal{C}}'[\vec{H'}^2]\ \overline{\lhd^{-1}}^{\mathbb{C}}\ \dot{\mathcal{C}}'[\vec{H'}^1]$, and hence:

$$\dot{N}_2 \dot{\preceq}_{Q'} \dot{\mathcal{C}}'[\vec{H'}^2]\ (\overline{\lhd}^{\mathbb{C}})^{-1}\ \dot{\mathcal{C}}'[\vec{H'}^1] \dot{\preceq}_{Q''} \dot{N}_1. \qquad \square$$

Appendix G. Sufficient conditions for robustness

A proof of robustness becomes trivial for a specimen with a rewrite token that gives a non-rooted state. Thanks to the lemma below, we can show that a state is not rooted, by checking paths from the token target.

Definition G.1 (Accessible paths)**.**

- A path of a hypernet is said to be *accessible* if it consists of edges whose all sources have type $\star$.
- An accessible path is called *stable* if the labels of its edges are included in $\{\mathsf{I}\} \cup \mathbb{O}_{\checkmark}$.
- An accessible path is called *active* if it starts with one active operation edge and possibly followed by a stable path.

Note that box edges and atom edges never appear in an accessible path.

Lemma G.2. *If a state has a rewrite token that is not an incoming edge of a contraction edge, then the state satisfies the following property: If there exists an accessible, but not active, path from the token target, then the state is not rooted.*

Proof. This is a contraposition of a consequence of Lemma C.5 and Lemma D.6(2). □

Checking the condition (III) of robustness (see Definition 8.8) involves finding a quasi-$\mathbb{C}$-specimen of $\lhd$ up to $(\dot{\preceq}_{Q'}, \dot{\preceq}_{Q''})$, namely checking the condition (B) of Definition 8.3(2). The following lemma enables us to use contextual refinement $\preceq^{\mathbb{C}}_{Q'}$ to yield state refinement $\dot{\preceq}_{Q'}$, via single $\mathbb{C}$-specimens of a certain pre-template $\lhd$.

Definition G.3. A pre-template $\lhd$ is a *trigger* if it satisfies the following:

(A) For any single $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\chi]; H^1; H^2)$ of $\lhd$, such that $\dot{\mathcal{C}}$ has an entering search token, $\dot{\mathcal{C}}[H^p] \to \langle \dot{\mathcal{C}}[H^p] \rangle_{\lightning/?}$ for each $p \in \{1, 2\}$.

(B) For any hypernets $H^1 \lhd H^2$, both H_1 and H_2 are one-way.

Lemma G.4. *Let $\mathbb{C}$ be a set of contexts, and Q' be a binary relation on $\mathbb{N}$ such that, for any $k_0, k_1, k_2 \in \mathbb{N}$, $(k_0 + k_1)\ Q'\ (k_0 + k_2)$ implies $k_1\ Q'\ k_2$. Let $\lhd$ be a pre-template that is a trigger and implies contextual refinement $\preceq^{\mathbb{C}}_{Q'}$. For any single $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\chi]; H^1; H^2)$ of $\lhd$, if compute transitions are all deterministic, and one of states $\dot{\mathcal{C}}[H^1]$ and $\dot{\mathcal{C}}[H^2]$ is rooted, then the other state is also rooted, and moreover, $\dot{\mathcal{C}}[H^1] \dot{\preceq}_{Q'} \dot{\mathcal{C}}[H^2]$.*

Proof. This is a corollary of Lemma G.5. □

Lemma G.5. *Let $\mathbb{C}$ be a set of contexts, and Q' be a binary relation on $\mathbb{N}$ such that, for any $k_0, k_1, k_2 \in \mathbb{N}$, $(k_0 + k_1)\ Q'\ (k_0 + k_2)$ implies $k_1\ Q'\ k_2$. Let $\lhd$ be a pre-template that is a trigger and implies contextual refinement $\preceq^{\mathbb{C}}_{Q'}$. For any single $\mathbb{C}$-specimen $(\dot{\mathcal{C}}[\chi]; H^1; H^2)$ of $\lhd$, the following holds.*

1. *For any $k \in \mathbb{N}$, $?; |\dot{\mathcal{C}}|[H^1] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^1]$ if and only if $?; |\dot{\mathcal{C}}|[H^2] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^2]$.*
2. *If compute transitions are all deterministic, and one of states $\dot{\mathcal{C}}[H^1]$ and $\dot{\mathcal{C}}[H^2]$ is rooted, then the other state is also rooted, and moreover, $\dot{\mathcal{C}}[H^1] \dot{\preceq}_{Q'} \dot{\mathcal{C}}[H^2]$.*

Proof of the point (1). Let (p, q) be an arbitrary element of a set $\{(1, 2), (2, 1)\}$. We prove that, for any $k \in \mathbb{N}$, $?; |\dot{\mathcal{C}}|[H^p] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^p]$ implies $?; |\dot{\mathcal{C}}|[H^q] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^q]$. The proof is by case analysis on the number k.

- When $k = 0$, $\dot{\mathcal{C}}[H^p]$ is initial, and by Lemma F.5(1), $\dot{\mathcal{C}}[H^q]$ is also initial. Note that $\lhd$ is a trigger and hence output-closed.
- When $k > 0$, by the following internal lemma, $?; |\dot{\mathcal{C}}|[H^q] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^q]$ follows from $?; |\dot{\mathcal{C}}|[H^p] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^p]$. □

Lemma G.6. *For any $m \in \{0, \dots, k\}$, there exists a focussed context $\dot{\mathcal{C}}'[\chi]$ such that $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}|$ and the following holds:*

$$?; |\dot{\mathcal{C}}|[H^p] \xrightarrow{\bullet}{}^{m} \dot{\mathcal{C}}'[H^p] \xrightarrow{\bullet}{}^{k-m} \dot{\mathcal{C}}[H^p],$$
$$?; |\dot{\mathcal{C}}|[H^q] \xrightarrow{\bullet}{}^{m} \dot{\mathcal{C}}'[H^q].$$

Proof. By induction on m. In the base case, when $m = 0$, we can take $?; |\dot{\mathcal{C}}|$ as $\dot{\mathcal{C}}'$.

In the inductive case, when $m > 0$, by induction hypothesis, there exists a focussed context $\dot{\mathcal{C}}'[\chi]$ such that $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}|$ and the following holds:

$$?; |\dot{\mathcal{C}}|[H^p] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}}'[H^p] \xrightarrow{\bullet}{}^{k-m+1} \dot{\mathcal{C}}[H^p],$$
$$?; |\dot{\mathcal{C}}|[H^q] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}}'[H^q].$$

Because $|\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}| \in \mathbb{C}$, $(\dot{\mathcal{C}}'; H^1; H^2)$ is a single $\mathbb{C}$-specimen of $\lhd$, which yields rooted states. Because $k - m + 1 > 0$, $\dot{\mathcal{C}}'$ cannot have a rewrite token. The rest of the proof is by case analysis on the token of $\dot{\mathcal{C}}'$.

- When $\dot{\mathcal{C}}'$ has an entering search token, because $\lhd$ is a trigger, $\dot{\mathcal{C}}'[H^r] \to \langle \dot{\mathcal{C}}'[H^r] \rangle_{\lightning/?}$ for each $r \in \{p, q\}$. Because $\langle \dot{\mathcal{C}}'[H^r] \rangle_{\lightning/?} = \langle \dot{\mathcal{C}}' \rangle_{\lightning/?}[H^r]$, and search transitions are deterministic, we have the following:

$$?; |\dot{\mathcal{C}}|[H^p] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}}'[H^p] \xrightarrow{\bullet} \langle \dot{\mathcal{C}}' \rangle_{\lightning/?}[H^p] \xrightarrow{\bullet}{}^{k-m} \dot{\mathcal{C}}[H^p],$$
$$?; |\dot{\mathcal{C}}|[H^q] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}}'[H^q] \xrightarrow{\bullet} \langle \dot{\mathcal{C}}' \rangle_{\lightning/?}[H^q].$$

  We also have $|\langle \dot{\mathcal{C}}' \rangle_{\lightning/?}| = |\dot{\mathcal{C}}'| = |\dot{\mathcal{C}}|$.
- When $\dot{\mathcal{C}}'$ has a value token, or a non-entering search token, because $\lhd$ is output-closed, by Lemma F.5(3), there exists a focussed context $\dot{\mathcal{C}}''$ such that $|\dot{\mathcal{C}}''| = |\dot{\mathcal{C}}'|$ and $\dot{\mathcal{C}}'[H^r] \to \dot{\mathcal{C}}''[H^r]$ for each $r \in \{p, q\}$. The transition $\dot{\mathcal{C}}'[H^r] \to \dot{\mathcal{C}}''[H^r]$, for each $r \in \{p, q\}$, is a search transition, and by the determinism of search transitions, we have the following:

$$?; |\dot{\mathcal{C}}|[H^p] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}}'[H^p] \xrightarrow{\bullet} \dot{\mathcal{C}}''[H^p] \xrightarrow{\bullet}{}^{k-m} \dot{\mathcal{C}}[H^p],$$
$$?; |\dot{\mathcal{C}}|[H^q] \xrightarrow{\bullet}{}^{m-1} \dot{\mathcal{C}}'[H^q] \xrightarrow{\bullet} \dot{\mathcal{C}}''[H^q].$$ □

Proof of the point (2). If one of states $\dot{\mathcal{C}}[H^1]$ and $\dot{\mathcal{C}}[H^2]$ is rooted, by the point (1), the other state is also rooted, and moreover, there exists $k \in \mathbb{N}$ such that $?; |\dot{\mathcal{C}}|[H^r] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^r]$ for each $r \in \{1, 2\}$.

Our goal is to prove that, for any $k_1 \in \mathbb{N}$ and any final state $\dot{N}_1$ such that $\dot{\mathcal{C}}[H^1] \to^{k_1} \dot{N}_1$, there exist $k_2 \in \mathbb{N}$ and a final state $\dot{N}_2$ such that $k_1\ Q'\ k_2$ and $\dot{\mathcal{C}}[H^2] \to^{k_2} \dot{N}_2$. Assuming $\dot{\mathcal{C}}[H^1] \to^{k_1} \dot{N}_1$, we have the following:

$$?; |\dot{\mathcal{C}}|[H^1] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^1] \to^{k_1} \dot{N}_1,$$
$$?; |\dot{\mathcal{C}}|[H^2] \xrightarrow{\bullet}{}^{k} \dot{\mathcal{C}}[H^2].$$

Because $\lhd$ implies contextual refinement $\preceq^{\mathbb{C}}_{Q'}$, and $|\dot{\mathcal{C}}| \in \mathbb{C}$, we have state refinement $?; |\dot{\mathcal{C}}|[H^1] \mathrel{\dot{\preceq}_{Q'}} ?; |\dot{\mathcal{C}}|[H^2]$. Therefore, there exist $l_2 \in \mathbb{N}$ and a final state $\dot{N}_2$ such that $(k + k_1)\ Q'\ l_2$ and $?; |\dot{\mathcal{C}}|[H^2] \to^{l_2} \dot{N}_2$.

The assumption that compute transitions are all deterministic implies that all transitions, including intrinsic ones, are deterministic. Following from this are $l_2 \geq k$ and the following:

$$?; |\dot{\mathcal{C}}|[H^1] \overset{\bullet}{\to}^{k} \dot{\mathcal{C}}[H^1] \to^{k_1} \dot{N}_1,$$
$$?; |\dot{\mathcal{C}}|[H^2] \overset{\bullet}{\to}^{k} \dot{\mathcal{C}}[H^2] \to^{l_2 - k} \dot{N}_2.$$

By the assumption on Q', $(k + k_1)\ Q'\ l_2$ implies $k_1\ Q'\ (l_2 - k)$. □

Appendix H. Local rewrite rules and transfer properties

The sufficiency-of-robustness theorem reduces a proof of an observational equivalence down to establishing robust templates. As illustrated in Section 12, this typically boils down to checking input-safety of pre-templates, and checking robustness of pre-templates relative to rewrite transitions.

The key part of checking input-safety or robustness of a pre-template is to analyse how a rewrite transition involves edges (at any depth) of a state that are contributed by the pre-template. In this section, we focus on rewrite transitions that are locally specified by means of the contraction rules or rewrite rules (e.g. the micro-beta rewrite rules), and identify some situation where these transitions involve the edges contributed by a pre-template in a *safe* manner. These situations can be formalised for arbitrary instances of the universal abstract machine, including the particular instance $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$ that is used in Section 12 to prove the Parametricity law.

This section proceeds as follows. Firstly, Appendix H.1 formalises the safe involvement in terms of *transfer* properties. Appendix H.2 establishes transfer properties for the particular pre-templates and local rewrite rules used in Section 12 to prove the Parametricity law. Finally, Appendix H.3 demonstrates the use of the transfer properties, by providing details of checking input-safety and robustness to prove the Parametricity law.

H.1. Transfer properties.

We refer to the contraction rules which locally specify copy transitions, and rewrite rules that locally specify rewrite transitions for active operations, altogether as *$\lightning$-rules*. To analyse how a $\lightning$-rule involves edges contributed by a pre-template, one would first need to check all possible overlaps between the local rule and the edges, and then observe how these overlaps are affected by application of the local rule. We identify *safe* involvement of the pre-template in the $\lightning$-rule, as the situation where the overlaps get only eliminated or duplicated without any internal modification.

We will first formalise safe involvement for a single application of $\lightning$-rules, and then for a pair of applications of $\lightning$-rules. The latter can capture safe involvement of edges contributed by a pre-template, which can be exploited to check input-safety and robustness of pre-templates.

Notation H.1. Let $m \in \mathbb{N}$ and $m' \in \mathbb{N}$. Given a sequence $\vec{x} = x_1, \ldots, x_m$ of length m and a function $f\colon \{1, \ldots, m'\} \to \{1, \ldots, m\}$, a sequence $f(\vec{x}) = x'_1, \ldots, x'_{m'}$ of length m' is given by $x'_j = x_{f(j)}$ for each $j \in \{1, \ldots, m'\}$.

Definition H.2 (Transfer of hypernets)**.** Let $\mathbb{C}$ and $\mathbb{C}'$ be two sets of focus-free contexts, and $\mathbb{H}$ be a set of focus-free hypernets. A $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ of a universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ *transfers* $\mathbb{H}$ *from* $\mathbb{C}$ *to* $\mathbb{C}'$ if, for any $m \in \mathbb{N}$, any focussed context $\dot{\mathcal{C}}[\chi_1, \ldots, \chi_m]$ such that $|\dot{\mathcal{C}}| \in \mathbb{C}$, and any m focus-free hypernets $G_i \in \mathbb{H}$ ($i \in \{1, \ldots, m\}$) such that $\dot{N} = \dot{\mathcal{C}}[G_1, \ldots, G_m]$, there exist some $m' \in \mathbb{N}$, some focussed context $\dot{\mathcal{C}}'[\chi'_1, \ldots, \chi'_{m'}]$, and some function $f\colon \{1, \ldots, m'\} \to \{1, \ldots, m\}$, and the following holds.

- $|\dot{\mathcal{C}}'| \in \mathbb{C}'$.
- $\dot{N}' = \dot{\mathcal{C}}'[f(G_1, \ldots, G_m)]$.
- $\dot{\mathcal{C}}[H_1, \ldots, H_m] \mapsto \dot{\mathcal{C}}'[f(H_1, \ldots, H_m)]$ is a $\lightning$-rule, for any m focus-free hypernets $H_i \in \mathbb{H}$ ($i \in \{1, \ldots, m\}$).

This transfer property enjoys monotonicity in the following sense: if a $\lightning$-rule transfers $\mathbb{H}$ from $\mathbb{C}$ to $\mathbb{C}'$, and $\mathbb{C}' \subseteq \mathbb{C}''$, then the $\lightning$-rule transfers $\mathbb{H}$ from $\mathbb{C}$ to $\mathbb{C}''$ as well. If a $\lightning$-rule transfers $\mathbb{H}$ from $\mathbb{C}$ to the same $\mathbb{C}$, we say the $\lightning$-rule *preserves* $\mathbb{H}$ *in* $\mathbb{C}$.

Given an operation set $\mathbb{O}$, we will be particularly interested in the following sets of hypernets and contexts for $\mathbb{O}$, some of which have already been introduced elsewhere: the set $\mathbb{H}_{\mathbb{O}}$ of all focus-free hypernets, the set $\mathbb{H}_{\otimes}$ of contraction trees, the set $\mathbb{C}_{\mathbb{O}}$ of all focus-free contexts, the set $\mathbb{C}_{\mathbb{O}\text{-bf}}$ of all binding-free contexts, the set $\mathbb{C}_{\mathbb{O}\text{-dp}}$ of all *deep* contexts, i.e. focus-free contexts whose holes are all deep.

Example H.3 (Transfer/preservation of hypernets in contexts)**.**

- When a $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ preserves $\mathbb{H}_{\mathbb{O}}$ in $\mathbb{C}_{\mathbb{O}\text{-dp}}$, any deep edge of $\dot{N}$ also appears as a deep edge in $\dot{N}'$, and it also retains its neighbours. This is trivially the case if the $\lightning$-rule involves no box edges (and hence deep edges) at all. It is also the case if the $\lightning$-rule only eliminates or duplicates box edges without modifying deep edges. The contraction rules are an example of duplicating boxes.
- Preservation of deep edges can be restricted to binding-free positions, which are specified by binding-free contexts. When a $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ preserves $\mathbb{H}_{\mathbb{O}}$ in $\mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$, any deep edge of $\dot{N}$ in a binding-free position also appears as a deep edge in a binding-free position in $\dot{N}'$.
- When a $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ transfers $\mathbb{H}_{\mathbb{O}}$ from $\mathbb{C}_{\mathbb{O}\text{-dp}}$ to $\mathbb{C}_{\mathbb{O}}$, any deep edge of $\dot{N}$ also appears as an edge in $\dot{N}'$, retaining its neighbours, but not necessarily as a deep edge. This is preservation of deep edges in a weak sense. It is the case when a $\lightning$-rule replaces a box edge with its contents, turning some deep edges into shallow edges without modifying their connection. The micro-beta rewrite rules are an example of this situation.
- When a $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ preserves $\mathbb{H}_{\otimes}$ in $\mathbb{C}_{\mathbb{O}}$, any contraction tree in $\dot{N}$ also appears in $\dot{N}'$. The contraction rules are designed to satisfy this preservation property. ■

Definition H.4 (Transfer of (rooted) specimens)**.** Let $\mathbb{C}$ and $\mathbb{C}'$ be two sets of focus-free contexts, and $\lhd$ be a pre-template.

- A $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ of a universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$ *transfers specimens of* $\lhd$ *from* $\mathbb{C}$ *to* $\mathbb{C}'$ if, for any $\mathbb{C}$-specimen of the form $(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2[\vec{\chi}'']]; \vec{G}', \vec{G}''; \vec{H}', \vec{H}'')$ such that $\dot{N} = \dot{\mathcal{C}}_2[\vec{G}'']$, there exist some focussed context $\dot{\mathcal{C}}'_2$ and two sequences $\vec{G}'''$ and $\vec{H}'''$ of focus-free hypernets, and the following holds.
  - $\dot{N}' = \dot{\mathcal{C}}'_2[\vec{G}''']$.
  - $\dot{\mathcal{C}}_2[\vec{H}''] \mapsto \dot{\mathcal{C}}'_2[\vec{H}''']$ is a $\lightning$-rule.

– $(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2']; \vec{G}', \vec{G}''; \vec{H}', \vec{H}'')$ is a $\mathbb{C}'$-specimen of $\lhd$.

- The $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ is said to transfer *rooted* specimens of $\lhd$ from $\mathbb{C}$ to $\mathbb{C}'$ if, in the above definition, the $\mathbb{C}$-specimen $(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2[\vec{\chi}'']]; \vec{G}', \vec{G}''; \vec{H}', \vec{H}'')$ is restricted to yield two rooted states $\mathcal{C}_1[\vec{G}', \dot{\mathcal{C}}_2[\vec{G}'']]$ and $\mathcal{C}_1[\vec{H}', \dot{\mathcal{C}}_2[\vec{H}'']]$.

If a $\lightning$-rule transfers specimens of $\lhd$ from $\mathbb{C}$ to the same $\mathbb{C}$, we say the $\lightning$-rule *preserves specimens of* $\lhd$ *in* $\mathbb{C}$.

We can prove that certain transfer properties of hypernets imply the corresponding transfer properties of specimens, as stated in Proposition H.6 below. These are primarily transfer of deep edges, and preservation of contraction trees. Proposition H.6 below will simplify some part of establishing input-safety and robustness of a pre-template, because it enables us to analyse a single application of a $\lightning$-rule on a state, instead of a pair of applications of a $\lightning$-rule on two states induced by a specimen of the pre-template.

Definition H.5 (Root-focussed $\lightning$-rules)**.** A $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ is said to be *root-focussed* if it satisfies the following.

- $\dot{N}$ has only one input.
- $\dot{N} = \lightning; |\dot{N}|$ holds, i.e. the sole input of $\dot{N}$ coincides with the source of the token.
- Every output of $\dot{N}$ is reachable from the sole input of $\dot{N}$.

Proposition H.6. *For any $\lightning$-rule $\dot{N} \mapsto \dot{N}'$ of a universal abstract machine $\mathcal{U}(\mathbb{O}, B_{\mathbb{O}})$, the following holds.*

(1) *If it transfers $\mathbb{H}_{\mathbb{O}}$ from $\mathbb{C}_{\mathbb{O}\text{-dp}}$ to $\mathbb{C}_{\mathbb{O}}$, it transfers specimens of any pre-template $\lhd$ from $\mathbb{C}_{\mathbb{O}\text{-dp}}$ to $\mathbb{C}_{\mathbb{O}}$.*
(2) *If it preserves $\mathbb{H}_{\mathbb{O}}$ in $\mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$, it preserves specimens of any pre-template $\lhd$ in $\mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$.*
(3) *If it is root-focussed and transfers $\mathbb{H}_{\mathbb{O}}$ from $\mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$ to $\mathbb{C}_{\mathbb{O}\text{-bf}}$, it transfers rooted specimens of any output-closed pre-template $\lhd$ from $\mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$ to $\mathbb{C}_{\mathbb{O}\text{-bf}}$.*
(4) *If it preserves $\mathbb{H}_{\otimes}$ in $\mathbb{C}_{\mathbb{O}}$, it preserves specimens of any pre-template $\lhd \subseteq \mathbb{H}_{\otimes} \times \mathbb{H}_{\otimes}$, which is on contraction trees, in $\mathbb{C}_{\mathbb{O}}$.*

Proof of the point (1). We take an arbitrary $\mathbb{C}_{\mathbb{O}\text{-dp}}$-specimen of the form

$$(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2[\vec{\chi}'']]; \vec{G}', \vec{G}''; \vec{H}', \vec{H}'')$$

of the pre-template $\lhd$, such that $\dot{N} = \dot{\mathcal{C}}_2[\vec{G}'']$. Because the context $\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2[\vec{\chi}'']]$ of the specimen must be focussed, the token in the context is shallow. This means that the hole labelled with χ in the context $\mathcal{C}_1[\vec{\chi}', \chi]$ must be shallow. On the other hand, the specimen satisfies $|\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2]| \in \mathbb{C}_{\mathbb{O}\text{-dp}}$, and hence $\mathcal{C}_1[\vec{\chi}', |\dot{\mathcal{C}}_2|] \in \mathbb{C}_{\mathbb{O}\text{-dp}}$. As a consequence, we have $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{O}\text{-dp}}$. Now, by the assumption, there exist some focussed context $\dot{\mathcal{C}}_2'$ and some function f, such that $|\dot{\mathcal{C}}_2'| \in \mathbb{C}_{\mathbb{O}}$ and $\dot{N}' = \dot{\mathcal{C}}_2'[f(\vec{G}'')]$ hold, and moreover, $\dot{\mathcal{C}}_2[\vec{H}''] \mapsto \dot{\mathcal{C}}_2'[f(\vec{H}'')]$ is also a $\lightning$-rule. We obtain a triple $(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2']; \vec{G}', f(\vec{G}''); \vec{H}', f(\vec{H}''))$. It satisfies $|\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2']| \in \mathbb{C}_{\mathbb{O}}$, and is a $\mathbb{C}_{\mathbb{O}}$-specimen of the pre-template $\lhd$. □

Proof of the point (2). We take an arbitrary $(\mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}})$-specimen of the form

$$(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2[\vec{\chi}'']]; \vec{G}', \vec{G}''; \vec{H}', \vec{H}'')$$

of the pre-template $\lhd$, such that $\dot{N} = \dot{\mathcal{C}}_2[\vec{G}'']$.

We first check that $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$ follows from $|\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2]| \in \mathbb{C}_{\mathbb{O}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}\text{-bf}}$, as follows.

- Because the context $\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}_2]$ must be focussed, the token in the context is shallow. This means that the hole labelled with χ in the context $\mathcal{C}_1[\vec{\chi'}, \chi]$ must be shallow. This, combined with $|\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}_2]| = \mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, implies $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}}$.
- If the context $|\dot{\mathcal{C}}_2|$ contains a path that makes it not binding-free, the path is also a path in the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|]$ and makes the context not binding-free. Therefore, because $|\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}_2]| = \mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|] \in \mathbb{C}_{\mathbb{0}\text{-bf}}$ holds, the context $|\dot{\mathcal{C}}_2|$ is without any path that makes the context not binding-free. This means $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{0}\text{-bf}}$.

By the assumption, there exist some focussed context $\dot{\mathcal{C}}'_2$ and some function f, such that $|\dot{\mathcal{C}}'_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}} \cap \mathbb{C}_{\mathbb{0}\text{-bf}}$ and $\dot{N}' = \dot{\mathcal{C}}'_2[f(\vec{G''})]$ hold, and moreover, $\dot{\mathcal{C}}_2[\vec{H''}] \mapsto \dot{\mathcal{C}}'_2[f(\vec{H''})]$ is also a $\lightning$-rule. We obtain a triple $(\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}'_2]; \vec{G'}, f(\vec{G''}); \vec{H'}, f(\vec{H''}))$.

To conclude the proof, it suffices to prove that this triple is a $(\mathbb{C}_{\mathbb{0}\text{-dp}} \cap \mathbb{C}_{\mathbb{0}\text{-bf}})$-specimen of the pre-template $\lhd$, which boils down to showing $|\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}'_2]| = \mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}} \cap \mathbb{C}_{\mathbb{0}\text{-bf}}$.

We firstly prove $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}}$. Because $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}}$ holds, the holes labelled with $\vec{\chi'}$ of the context $\mathcal{C}_1$ must be all deep. This, together with $|\dot{\mathcal{C}}'_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, implies $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}}$.

We then prove $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|] \in \mathbb{C}_{\mathbb{0}\text{-bf}}$ by contradiction, which will conclude the whole proof. Assume that the context is not binding-free. It has a path P from a source of an edge e that is either a contraction edge, an atom edge, a box edge or a hole edge, to a source of an edge e' that is a hole edge. Thanks to $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}}$ and $|\dot{\mathcal{C}}'_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, the hole edge e' of the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|]$ must be deep. We will infer a contradiction by case analysis on the hole edge e'. There are two cases.

- One case is when the edge e' of the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|]$ comes from the context $\mathcal{C}_1$. In this case, the edge e' is one of the deep hole edges labelled with $\vec{\chi'}$. This means that the path P in the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|]$ must consist of deep edges only, and these deep edges, together with the edge e', must be contained in a box of the context $\mathcal{C}_1$. Therefore the path P is also a path in the context $\mathcal{C}_1$, and it makes the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|]$ not binding-free. This contradicts $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|] \in \mathbb{C}_{\mathbb{0}\text{-bf}}$.
- The other case is when the edge e' of the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}'_2|]$ comes from the context $|\dot{\mathcal{C}}'_2|$. In this case, the edge e' is a hole edge of the context $|\dot{\mathcal{C}}'_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, and hence a deep edge. This means that the path P is also a path in the context $|\dot{\mathcal{C}}'_2|$, consisting of deep edges only. The path P therefore makes the context $|\dot{\mathcal{C}}'_2|$ not binding-free, which contradicts $|\dot{\mathcal{C}}'_2| \in \mathbb{C}_{\mathbb{0}\text{-bf}}$. □

Proof of the point (3). We take an arbitrary $(\mathbb{C}_{\mathbb{0}\text{-dp}} \cap \mathbb{C}_{\mathbb{0}\text{-bf}})$-specimen of the form

$$(\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}_2[\vec{\chi''}]]; \vec{G'}, \vec{G''}; \vec{H'}, \vec{H''})$$

of the pre-template $\lhd$, such that $\dot{N} = \dot{\mathcal{C}}_2[\vec{G''}]$ holds, and two states $\mathcal{C}_1[\vec{G'}, \dot{\mathcal{C}}_2[\vec{G''}]]$ and $\mathcal{C}_1[\vec{H'}, \dot{\mathcal{C}}_2[\vec{H''}]]$ are both rooted.

We can first check $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}} \cap \mathbb{C}_{\mathbb{0}\text{-bf}}$, in the same way as the proof of the point (2). By the assumption, there exist some focussed context $\dot{\mathcal{C}}'_2$ and some function f, such that $|\dot{\mathcal{C}}'_2| \in \mathbb{C}_{\mathbb{0}\text{-bf}}$ and $\dot{N}' = \dot{\mathcal{C}}'_2[f(\vec{G''})]$ hold, and moreover, $\dot{\mathcal{C}}_2[\vec{H''}] \mapsto \dot{\mathcal{C}}'_2[f(\vec{H''})]$ is also a $\lightning$-rule. We obtain a triple $(\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}'_2]; \vec{G'}, f(\vec{G''}); \vec{H'}, f(\vec{H''}))$.

To conclude the proof, it suffices to prove that this triple is a $\mathbb{C}_{\mathbb{0}\text{-bf}}$-specimen of the pre-template $\lhd$, which boils down to showing $|\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}_2']| = \mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|] \in \mathbb{C}_{\mathbb{0}\text{-bf}}$. We prove this by contradiction, as follows.

Assume that the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|]$ is not binding-free. It has a path P from a source of an edge e that is either a contraction edge, an atom edge, a box edge or a hole edge, to a source of an edge e' that is a hole edge. We will infer a contradiction by case analysis on the edge e'. There are two cases.

- One case is when the edge e' of the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|]$ comes from the context $\mathcal{C}_1$. In this case, the edge e' is one of the hole edges labelled with $\vec{\chi'}$. Because of $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|] \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, the hole edge e' must be deep. This means that the path P must consist of deep edges contained in a box of the context $\mathcal{C}_1$. The path is therefore a path in the context $\mathcal{C}_1$, and also in the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|]$. This means $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2|] \notin \mathbb{C}_{\mathbb{0}\text{-bf}}$, which is a contradiction.
- The other case is when the edge e' of the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|]$ comes from the context $|\dot{\mathcal{C}}_2'|$. In this case, we will infer a contradiction by further case analysis on the edge e and the path P. There are three (sub-)cases.
  - The first case is when the edge e comes from the context $|\dot{\mathcal{C}}_2'|$ and P is a path in the context. In this case, the path P makes the context $|\dot{\mathcal{C}}_2'|$ not binding-free, which contradicts $|\dot{\mathcal{C}}_2'| \in \mathbb{C}_{\mathbb{0}\text{-bf}}$.
  - The second case is when the edge e comes from the context $|\dot{\mathcal{C}}_2'|$ and P does not give a single path in the context. In this case, the edges e and e' both come from the context $|\dot{\mathcal{C}}_2'|$, but P is a valid path only in the whole context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|]$. This means that, in the context $\mathcal{C}_1[\vec{\chi'}, \chi]$, a source of the hole edge labelled with χ is reachable from a target of the same hole edge.

    Because the $\lightning$-rule $\dot{\mathcal{C}}_2[\vec{G''}] \mapsto \dot{\mathcal{C}}_2'[f(\vec{G''})]$ is root-focussed, the focussed hypernet $\dot{\mathcal{C}}_2[\vec{G''}]$ has only one input, the input coincides with the source of the token, and every output of the hypernet is reachable from the sole input. Moreover, because of $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, the same holds for the focussed context $\dot{\mathcal{C}}_2$ too, namely: the context has only one input, the input coincides with the source of the token, and every output of the context is reachable from the sole input.

    As a consequence, in the focussed context $\mathcal{C}_1[\vec{\chi'}, \dot{\mathcal{C}}_2]$, the token source is reachable from itself, via a cyclic path that contains some edges coming from the context $\dot{\mathcal{C}}_2$ including the token edge. This path is not an operation path. Therefore, by Lemma C.7(3), at least one of the states $\mathcal{C}_1[\vec{G'}, \dot{\mathcal{C}}_2[\vec{G''}]]$ and $\mathcal{C}_1[\vec{H'}, \dot{\mathcal{C}}_2[\vec{H''}]]$ is not rooted. This is a contradiction.
  - The last case is when the edge e comes from the context $\mathcal{C}_1$. Recall that the edge e' comes from the context $|\dot{\mathcal{C}}_2'|$. In this case, the path P in the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|]$ has a prefix that gives a path P' in the context $\mathcal{C}_1[\vec{\chi'}, \chi]$, from the same source of the edge e as the path P, to a source of the hole edge labelled with χ. Because the path P' is given as a part of the path P in the context $\mathcal{C}_1[\vec{\chi'}, |\dot{\mathcal{C}}_2'|]$, the path P' in the context $\mathcal{C}_1[\vec{\chi'}, \chi]$ does not itself contain the hole edge labelled with χ.

    Because the $\lightning$-rule $\dot{\mathcal{C}}_2[\vec{G''}] \mapsto \dot{\mathcal{C}}_2'[f(\vec{G''})]$ is root-focussed, the focussed hypernet $\dot{\mathcal{C}}_2[\vec{G''}]$ has only one input, and the input coincides with the source of the token. Moreover, because of $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{0}\text{-dp}}$, the same holds for the focussed context $\dot{\mathcal{C}}_2$ too, namely: the context has only one input, and the input coincides with the source of the token.

As a consequence, the path P' in turn gives a path in the focussed context $\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2]$, from the same source of the edge e as the path P, to the source of the token. The first edge e of this path is not an operation edge, and therefore the path is not an operation path. By Lemma C.7(3), at least one of the states $\mathcal{C}_1[\vec{G}', \dot{\mathcal{C}}_2[\vec{G''}]]$ and $\mathcal{C}_1[\vec{H}', \dot{\mathcal{C}}_2[\vec{H''}]]$ is not rooted. This is a contradiction. □

Proof of the point (4). We take an arbitrary $\mathbb{C}_{\mathbb{O}}$-specimen of the form

$$(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2[\vec{\chi''}]]; \vec{G}', \vec{G''}; \vec{H}', \vec{H''})$$

of the pre-template $\lhd \subseteq \mathbb{H}_{\otimes} \times \mathbb{H}_{\otimes}$, such that $\dot{N} = \dot{\mathcal{C}}_2[\vec{G''}]$. All the hypernets in $\vec{G}', \vec{G''}, \vec{H}', \vec{H''}$ are elements of $\mathbb{H}_{\otimes}$, i.e. contraction trees. It trivially holds that $|\dot{\mathcal{C}}_2| \in \mathbb{C}_{\mathbb{O}}$. Therefore, by the assumption, there exist some focussed context $\dot{\mathcal{C}}_2'$ and some function f, such that $|\dot{\mathcal{C}}_2'| \in \mathbb{C}_{\mathbb{O}}$ and $\dot{N}' = \dot{\mathcal{C}}_2'[f(\vec{G''})]$ hold, and moreover, $\dot{\mathcal{C}}_2[\vec{H''}] \mapsto \dot{\mathcal{C}}_2'[f(\vec{H''})]$ is also a $\lightning$-rule. We obtain a triple $(\mathcal{C}_1[\vec{\chi}', \dot{\mathcal{C}}_2']; \vec{G}', f(\vec{G''}); \vec{H}', f(\vec{H''}))$, and this is a $\mathbb{C}_{\mathbb{O}}$-specimen of the pre-template $\lhd$. □

H.2. **Transfer properties for the Parametricity law.** We can now establish transfer properties of deep edges and contraction trees for the particular machine $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$ which is used to prove the Parametricity law.

Proposition H.7. *The universal abstract machine $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$ satisfies the following.*

(1) *The contraction rules and all rewrite rules transfer $\mathbb{H}_{\mathbb{O}^{\mathrm{ex}}}$ from $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}}$ to $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$.*
(2) *The contraction rules, and all rewrite rules except for the micro-beta rewrite rules, preserve $\mathbb{H}^{\mathrm{ex}}_{\mathbb{O}}$ in $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$.*
(3) *The micro-beta rewrite rules transfer $\mathbb{H}^{\mathrm{ex}}_{\mathbb{O}}$ from $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$ to $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$.*
(4) *The contraction rules and all rewrite rules preserve $\mathbb{H}_{\otimes}$ in $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$.*

Sketch of the proof. We can prove the four points by analysing each $\lightning$-rule, i.e. a contraction rule or a local rewrite rule for an active operation, of the universal abstract machine $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$.

Firstly, the only way in which a contraction rule involves deep edges is to have them inside the hypernet to be duplicated (H in Figure 17). The deep edges and their connection are all preserved, and replacing these edges with arbitrary deep edges still enables the contraction rule. The point (1) therefore holds. Additionally, any path to a source of a deep edge must consist of deep edges only, and if such a path appears in the result $\dot{N}'$ of a contraction rule $\dot{N} \mapsto \dot{N}'$, the path necessarily appears in the original hypernet $\dot{N}$ too. Therefore, if the contraction rule moves deep edges out of binding-free positions, these edges must not be at binding-free positions beforehand. This is a contradiction, and the point (2) holds. As for contraction trees, whenever a contraction rule involves a contraction tree, the tree is either deep and gets duplicated, or shallow and left unmodified. Replacing the contraction tree with another contraction tree still enables the contraction rule that duplicates the same hypernet. The point (4) therefore holds.

Secondly, we analyse the micro-beta rewrite rules. Whenever deep edges are involved in a micro-beta rewrite rule, they must be inside the box edge that gets opened (i.e. G in Figure 10b). These deep edges may be turned into shallow edges, but their connection is unchanged. The difference of deep edges does not affect application of the rule, and hence the point (1) holds. If these deep edges are at binding-free positions, they remain at binding-free positions after applying the micro-beta rewrite rule, for a similar reason as the

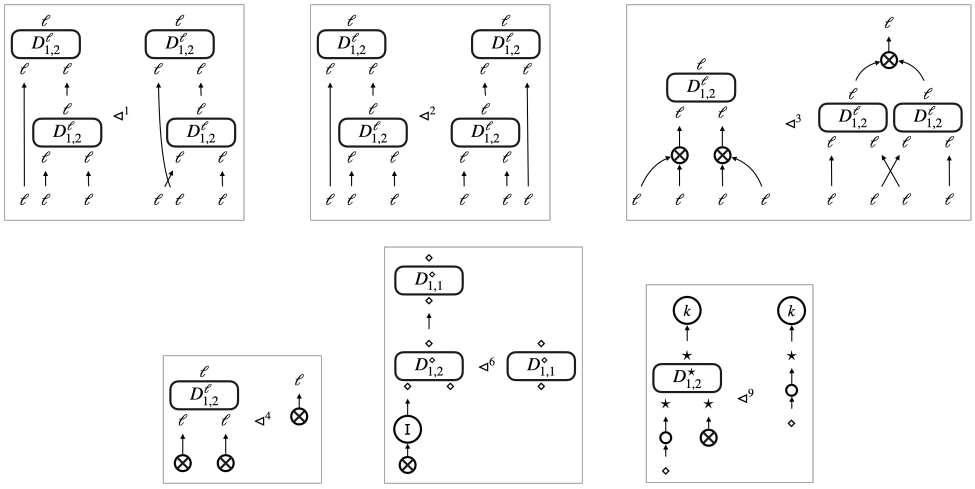


Figure 24: Triggers where $k \in \mathbb{N}$

contraction rules. The point (3) therefore holds. As for contraction trees, the only way in which contraction trees get involved in a micro-beta rewrite rule is for them to be deep. The point (4) reduces to the point (1) for micro-beta rewrite rules.

The rest of the local rewrite rules involve no deep edges at all, and therefore points (1) and (2) trivially hold. These rules either involve no contraction trees, or involve shallow contraction trees without any modification. The difference of contraction trees does not affect application of the rules. The point (4) therefore holds. □

Corollary H.8. *In the universal abstract machine* $\mathcal{U}(\mathbb{O}^{\mathrm{ex}}, B_{\mathbb{O}^{\mathrm{ex}}})$*, the contraction rules and all rewrite rules satisfies the following.*

(1) *The rules transfer specimens of any pre-template* $\lhd$ *from* $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}}$ *to* $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$*.*
(2) *The rules transfer rooted specimens of any output-closed pre-template* $\lhd$ *from* $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$ *to* $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$*.*
(3) *The rules preserve specimens of any pre-template* $\lhd \subseteq \mathbb{H}_{\otimes} \times \mathbb{H}_{\otimes}$*, which is on contraction trees, in* $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$*.*

Proof. This is a consequence of Proposition H.7 and Proposition H.6, noting that the micro-beta rewrite rules are root-focussed and preserving in $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$ implies transferring from $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-dp}} \cap \mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$ to $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$. □

H.3. **Input-safety and robustness for the Parametricity law.** In this section we give some details of proving input-safety and robustness of the pre-templates for the Parametricity law, as indicated in Table 3. The proofs exploit the transfer properties established in Corollary H.8.

Figure 24 lists triggers that we use to prove input-safety and robustness of some of the pre-templates[7]. Table 5 shows contextual refinements/equivalences implied by these triggers (in the "implication" column), given that some pre-templates (shown in the "dependency"

[7]The numbering of triggers is according to the one used in [Mur20, Section 4.5.5]. Some of triggers in *loc. cit.* are for observational equivalences that we do not consider in this paper, and hence not presented here.

	dependency	implication of $H_1 \lhd H_2$	used for
$\lhd^1$	$\lhd^{\otimes\mathrm{Assoc}}, \lhd^{\otimes\mathrm{Comm}}$	$H_1 \simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=_{\mathbb{N}}} H_2$	$\lhd^{\otimes}$
$\lhd^2$	$\lhd^{\otimes\mathrm{Assoc}}, \lhd^{\otimes\mathrm{Comm}}$	$H_1 \simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=_{\mathbb{N}}} H_2$	$\lhd^{\otimes}$
$\lhd^3$	$\lhd^{\otimes\mathrm{Assoc}}, \lhd^{\otimes\mathrm{Comm}}, \lhd^{\otimes\mathrm{Idem}}$	$H_1 \simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=_{\mathbb{N}}} H_2$	$\lhd^{\mathrm{BPullC}}$
$\lhd^4$	$\lhd^{\otimes\mathrm{Idem}}$	$H_1 \simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=_{\mathbb{N}}} H_2$	$\lhd^{\mathrm{BPullW}}$
$\lhd^6$	$\lhd^{\otimes\mathrm{Assoc}}, \lhd^{\otimes\mathrm{Idem}}, \lhd^{\mathrm{GC}}$	$H_1 \simeq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{=_{\mathbb{N}}} H_2$	$\lhd^{\mathrm{Param}}$
$\lhd^9$	$\lhd^{\otimes}, \lhd^{\mathrm{GC}}$	$H_1 \preceq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{\geq_{\mathbb{N}}} H_2,\ H_2 \preceq^{\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}}_{\leq_{\mathbb{N}}} H_1$	$\lhd^{\mathrm{Param}}$

Table 5: Triggers and their implied contextual refinements/equivalences

column) imply contextual refinement as shown in Table 3. All the implications can be proved simply using the congruence property and transitivity of contextual refinement. Table 5 shows which pre-template requires each trigger in its proof of input-safety or robustness (in the "used for" column). Note that the converse of any trigger is again a trigger.

Recall that there is a choice of contraction trees upon applying a contraction rule and some of the local rewrite rules. The minimum choice is to collect only contraction edges whose target is reachable from the token target. The maximum choice is to take the contraction tree(s) so that no contraction or weakening edge is incoming to the unique hole edge in a context.

H.3.1. *Pre-templates on contraction trees.* First we check input-safety and robustness of $\lhd^{\otimes\mathrm{Assoc}}$, $\lhd^{\otimes\mathrm{Comm}}$ and $\lhd^{\otimes\mathrm{Idem}}$, which are all on contraction trees.

Input-safety of $\lhd^{\otimes\mathrm{Assoc}}$ and $\lhd^{\otimes\mathrm{Comm}}$ can be checked as follows. Given a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen $(\dot{\mathcal{C}}; \vec{H^1}; \vec{H^2})$ with an entering search token, because any input of a contraction tree is a source of a contraction edge, we have:

$$\dot{\mathcal{C}}[\vec{H^1}] \xrightarrow{\bullet} \langle \dot{\mathcal{C}}[\vec{H^1}] \rangle_{\lightning/?}, \quad \dot{\mathcal{C}}[\vec{H^2}] \xrightarrow{\bullet} \langle \dot{\mathcal{C}}[\vec{H^2}] \rangle_{\lightning/?}.$$

It can be observed that a rewrite transition is possible in $\langle \dot{\mathcal{C}}[\vec{H^1}] \rangle_{\lightning/?}$ if and only if a rewrite transition is possible in $\langle \dot{\mathcal{C}}[\vec{H^2}] \rangle_{\lightning/?}$. When a rewrite transition is possible in both states, we can use Corollary H.8(3), by considering a maximal possible contraction rule. The results of the rewrite transition can be given by a new quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, =)$ (here $=$ denotes equality on states). When no rewrite transition is possible, both of the states are not final but stuck.

Robustness of the three pre-templates and their converse can also be proved using Corollary H.8(3), by considering a maximal possible local (contraction or rewrite) rule in each case.

H.3.2. *Input-safety of pre-templates not on contraction trees.* As mentioned in Section 12.2, pre-templates that relate hypernets with no input of type $\star$ are trivially input-safety for any parameter $(\mathbb{C}, Q, Q')$. This leaves us pre-templates $\lhd^{\otimes}$, $\lhd^{\vec{@}}$, $\lhd^{\mathtt{ref}}$ and $\lhd^{\mathrm{Param}}$ to check.

As for $\lhd^{\otimes}$, note that the pre-template $\lhd^{\otimes}$ relates hypernets with at least one input. Any $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen of $\lhd^{\otimes}$ with an entering search token can be turned into the form $(\mathcal{C}[(?;_j \chi), \vec{\chi}]; H^1, \vec{H^1}; H^2, \vec{H^2})$ where j is a positive number. The proof is by case analysis on the number j.

- When $j = 1$, we have:
$$\begin{aligned}\mathcal{C}[(?;_j H^1), \vec{H^1}] &\xrightarrow{\bullet} \mathcal{C}[(\lightning;_j H^1), \vec{H^1}]\\ &\to \mathcal{C}[(?;_j H^2), \vec{H^1}].\end{aligned}$$
We can take $(\mathcal{C}[(?;_j H^2), \vec{\chi}]; \vec{H^1}; \vec{H^2})$ as a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen, and the token in $\mathcal{C}[(?;_j H^2), \vec{\chi}]$ is not entering.
- When $j > 1$, the token target must be a source of a contraction edge. There exist a focus-free context $\mathcal{C}'[\chi']$, two focus-free hypernets $H'^1 \lhd^{\otimes} H'^2$ and a focus-free hypernet G, such that
$$\begin{aligned}\mathcal{C}[(?;_j H^1), \vec{H^1}] &\xrightarrow{\bullet} \mathcal{C}[(\lightning;_j H^1), \vec{H^1}]\\ &\to \mathcal{C}[(?;_j \mathcal{C}'[H'^1]), \vec{H^1}],\\ \mathcal{C}[(?;_j H^2), \vec{H^2}] &\xrightarrow{\bullet} \mathcal{C}[(\lightning;_j H^2), \vec{H^2}]\\ &\to \mathcal{C}[(?;_j G), \vec{H^2}],\end{aligned}$$
and $\mathcal{C}'[H'^2] \dot{\simeq}_{=_{\mathbb{N}}} G$ given by the trigger $\lhd^1$ via Lemma G.4. The results of these sequences give a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, \dot{\simeq}_{=_{\mathbb{N}}})$.

A proof of input-safety of the operational pre-templates $\lhd^{\vec{@}}$ and $\lhd^{\mathtt{ref}}$ is a simpler version of that of $\lhd^{\otimes}$, because the operational pre-templates relate hypernets with only one input.

Let $\mathbb{C}$ be either $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$ or $\mathbb{C}_{\mathbb{O}^{\mathrm{ex\text{-}bf}}}$. Any $\mathbb{C}$-specimen of an operational pre-template with an entering search token can be turned into the form $(\mathcal{C}[(?;\chi), \vec{\chi}]; H^1, \vec{H^1}; H^2, \vec{H^2})$; note that the parameter j that we had for $\lhd^{\otimes}$ is redundant in $?;\chi$. We have:
$$\begin{aligned}\mathcal{C}[(?; H^1), \vec{H^1}] &\xrightarrow{\bullet} \mathcal{C}[(\lightning; H^1), \vec{H^1}]\\ &\to \mathcal{C}[(?; H^2), \vec{H^1}].\end{aligned}$$
We can take $(\mathcal{C}[(?; H^2), \vec{\chi}]; \vec{H^1}; \vec{H^2})$ as a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen, and the token in $\mathcal{C}[(?;_j H^2), \vec{\chi}]$ is not entering. This data gives a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex\text{-}bf}}}$-specimen when $\mathbb{C} = \mathbb{C}_{\mathbb{O}^{\mathrm{ex\text{-}bf}}}$, which follows from the closedness of $\mathbb{C}_{\mathbb{O}^{\mathrm{ex\text{-}bf}}}$ with respect to plugging (Lemma E.1). Note that $?; H^1$ can be seen as a context with no holes, which is trivially binding-free.

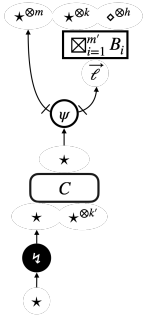
Finally, we look at the parametricity pre-template $\lhd^{\mathrm{Param}}$. Any $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen of this pre-template, with an entering search token, can be turned into the form $(\mathcal{C}[(?;_j \chi), \vec{\chi}]; H^1, \vec{H^1}; H^2, \vec{H^2})$ where j is a positive number. The token target is a source of an edge labelled with $\lambda \in \mathbb{O}^{\mathrm{ex}}_{\checkmark}$, so we have:
$$\begin{aligned}\mathcal{C}[(?;_j H^1), \vec{H^1}] &\xrightarrow{\bullet} \mathcal{C}[(\checkmark;_j H^1), \vec{H^1}],\\ \mathcal{C}[(?;_j H^2), \vec{H^2}] &\xrightarrow{\bullet} \mathcal{C}[(\checkmark;_j H^2), \vec{H^2}].\end{aligned}$$

The results of these sequences give a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=,=)$.

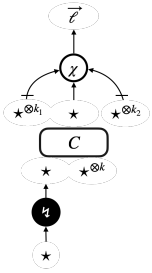
H.3.3. *Robustness of pre-templates not on contraction trees: a principle.* Robustness can be checked by inspecting rewrite transition $\dot{\mathcal{C}}[\vec{H}^1] \to \dot{N}'$ from the state given by a specimen $(\dot{\mathcal{C}}; \vec{H}^1; \vec{H}^2)$ of a pre-template, where the token of $\dot{\mathcal{C}}$ is not entering. We in particular consider the minimum local (contraction or rewrite) rule $\dot{G} \mapsto \dot{G}'$ applied in this transition. This means that, in the hypernet $\dot{G}$, every vertex is reachable from the token target.

The inspection boils down to analyse how the minimum local rule involves edges that come from the hypernets $\vec{H}^1$. If all the involvement is deep, i.e. only deep edges from $\vec{H}^1$ are involved in the local rule, these deep edges must come via deep holes in the context $\dot{\mathcal{C}}$. We can use Corollary H.8(1).

If the minimum local rule involves shallow edges that are from $\vec{H}^1$, endpoints of these edges are reachable from the token target. This means that, in the context $\dot{\mathcal{C}}$, some holes are shallow and their sources are reachable from the token target. Moreover, given that the token is not entering in $\dot{\mathcal{C}}$, the context has a path from the token target to a source of a hole edge.



(a) A shallow overlap where C is a contraction tree and B_i are box edges



(b) A focussed context where C is a contraction tree

For example, in checking robustness of $\lhd^{\mathrm{BPerm}}$ with respect to copy transitions, one situation of shallow overlaps is when $\dot{G}$ is in the form in Figure 25a, and some of the box edges B_i are from $\vec{H}^1$. Taking the minimum contraction rule means that C in the graph is a contraction tree that gives a path from the token target. This path C followed by the operation edge ϕ corresponds to paths from the token target to hole sources in the context $\dot{\mathcal{C}}$.

So, if the minimum local rule involves shallow edges that are from $\vec{H}^1$, the context $\dot{\mathcal{C}}$ necessarily has a path P from the token target to a hole source. The path becomes a path in the state $\dot{\mathcal{C}}[\vec{H}^1]$, from the token target to a source of an edge e that is from $\vec{H}^1$. The edge e is necessarily shallow, and also involved in the application of the minimum local rule, because of the connectivity of $\dot{G}$. Moreover, a source of the edge e is an input, in the relevant hypernet of $\vec{H}^1$. By inspecting minimum local rules, we can enumerate possible labelling of

the path P and the edge e, as summarised in Table 6. Explanation on the notation used in the table is to follow.

local rule	labels of path P	label of edge e
contraction	$(\otimes^\star_{\mathsf{C}})^+ \cdot \mathbb{O}^{\mathrm{ex}}$	box
	$(\otimes^\star_{\mathsf{C}})^+$	$\otimes^\star_{\mathsf{C}},\ \mathsf{I},\ \mathbb{O}^{\mathrm{ex}}$
$\overrightarrow{@}$	$\overrightarrow{@} \cdot \lambda$	box
	$\overrightarrow{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^*$	$\mathbb{O}^{\mathrm{ex}}_{\checkmark},\ \mathsf{I}$
$\mathtt{ref}$	$\mathtt{ref} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^*$	$\mathbb{O}^{\mathrm{ex}}_{\checkmark},\ \mathsf{I}$
$=$	$=$	I
	$= \cdot\, \mathsf{I} \cdot (\otimes^\diamond_{\mathsf{C}})^*$	$\otimes^\diamond_{\mathsf{C}},\ \circ$
$:=$	$:= \cdot\, (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^* \cdot \mathsf{I} \cdot (\otimes^\diamond_{\mathsf{C}})^*$	$\otimes^\diamond_{\mathsf{C}},\ \circ$
	$:= \cdot\, (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^*$	$\mathbb{O}^{\mathrm{ex}}_{\checkmark},\ \mathsf{I}$
$!$	$!$	I
	$! \cdot \mathsf{I} \cdot (\otimes^\diamond_{\mathsf{C}})^*$	$\otimes^\diamond_{\mathsf{C}},\ \circ$
$+$	$+$	$\mathbb{N}$
$-$	$-$	$\mathbb{N}$
$-_1$	$-_1$	$\mathbb{N}$

Table 6: Summary of paths that witness shallow overlaps

We use the regular-expression like notation in Table 6. For example, $(\otimes^\star_{\mathsf{C}})^+ \cdot \mathbb{O}^{\mathrm{ex}}$ represents finite sequences of edge labels, where more than one occurrences of the label $\otimes^\star_{\mathsf{C}}$ is followed by one operation $\phi \in \mathbb{O}^{\mathrm{ex}}$. This characterises paths that inhabit the overlap shown in Figure 25a, i.e. the contraction tree C followed by the operation edge ϕ. Note that this regular-expression like notation is not a proper regular expression, because it is over the infinite alphabet $M_{\mathbb{O}^{\mathrm{ex}}}$, the edge label set, and it accordingly admits infinite alternation (aka. union) implicitly.

To wrap up, checking robustness of each pre-template that is not on contraction trees boils down to using Corollary H.8(1) and/or analysing the cases enumerated in the table above.

H.3.4. *Robustness of $\lhd^\otimes$ and its converse.* Robustness check of the pre-template $\lhd^\otimes$ with respect to copy transitions has two cases. The first case is when one shallow overlap is caused by a path characterised by $(\otimes^\star_{\mathsf{C}})^+$, and the second case is when no shallow overlaps are present and Corollary H.8(1) can be used.

In the first case, namely, a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen with a non-entering rewrite token can be turned into the form $(\mathcal{C}[\dot{\mathcal{C}}'[\chi'], \vec{\chi}]; H^1, \vec{H^1}; H^2, \vec{H^2})$ where j is a positive number, and $\dot{\mathcal{C}}'$ is a focussed context in the form of Figure 25b. A rewrite transition is possible on both states given by the specimen, in which a contraction rule is applied to $\dot{\mathcal{C}}'[H^1]$ and $\dot{\mathcal{C}}'[H^2]$. Results of the rewrite transition give a new quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen. When $k_1 = 0$, this quasi-specimen is up to $(=, \dot{\simeq}_{=_{\mathbb{N}}})$, using the trigger $\lhd^2$. When $k_1 > 0$, the quasi-specimen is also up to $(=, \dot{\simeq}_{=_{\mathbb{N}}})$, but using the trigger $\lhd^1$.

Robustness check of the pre-template $\lhd^{\otimes}$ with respect to rewrite transitions always boils down to Corollary H.8(1). This is intuitively because no local rewrite rule of operations involves any shallow contraction edge of type $\star$.

Robustness of $(\lhd^{\otimes})^{-1}$ can be checked in a similar manner. Namely, using Table 6, shallow overlaps are caused by paths:

$(\otimes^{\star}_{\mathsf{C}})^{+}$,	$=$,	$+$,
$\overrightarrow{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\swarrow})^{*}$,	$:= \cdot (\mathbb{O}^{\mathrm{ex}}_{\swarrow})^{*}$,	$-$,
$\mathtt{ref} \cdot (\mathbb{O}^{\mathrm{ex}}_{\swarrow})^{*}$,	$!$,	$-_1$

from the token target. All paths but $(\otimes^{\star}_{\mathsf{C}})^{+}$ gives rise to a state that is not rooted, which can be checked using Lemma G.2. This reduces the robustness check of $(\lhd^{\otimes})^{-1}$ to that of $\lhd^{\otimes}$.

H.3.5. *Robustness of* $\lhd^{\mathrm{GC}}$ *and its converse.* These two pre-templates both relate hypernets with no inputs. Proofs of their robustness always boil down to the use of Corollary H.8(1), following the discussion in Section H.3.3. Namely, it is impossible to find the path P in the context $\dot{\mathcal{C}}$ from the token target to a hole source.

H.3.6. *Robustness of* $\lhd^{\mathrm{BPerm}}$, $\lhd^{\mathrm{BPullC}}$ *and* $\lhd^{\mathrm{BPullW}}$, *and their converse.* These six pre-templates all concern boxes. Using Table 6, shallow overlaps are caused by paths $(\otimes^{\star}_{\mathsf{C}})^{+} \cdot \mathbb{O}^{\mathrm{ex}}$ and $\overrightarrow{@} \cdot \lambda$ from the token target.

Robustness check with respect to compute transitions of operations $\mathbb{O}^{\mathrm{ex}}_{\nnearrow} \backslash \{\overrightarrow{@}\}$ always boil down to Corollary H.8(1).

As for compute transitions of the operation '$\overrightarrow{@}$', either one path $\overrightarrow{@} \cdot \lambda$ causes one shallow overlap, or all overlaps are deep. The latter situation boils down to Corollary H.8(1). In the former situation, a micro-beta rule involves one box that is contributed by a pre-template, and states given by a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen are turned into a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, =)$, by one rewrite transition.

As for copy transitions, there are two possible situations.

- Paths $(\otimes^{\star}_{\mathsf{C}})^{+} \cdot \mathbb{O}^{\mathrm{ex}}$ cause some shallow overlaps and there are some deep overlaps too.
- All overlaps are deep, which boils down to Corollary H.8(1).

In the first situation, some of the shallow boxes duplicated by a contraction rule are contributed by a pre-template, and other duplicated boxes may have deep edges contributed by the pre-template. By tracking these shallow and deep contributions in a contraction rule, it can be checked that one rewrite transition turns states given by a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen into a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen. This quasi-specimen is up to the following, depending on pre-templates:

- $(=, =)$ for $\lhd^{\mathrm{BPerm}}$ and its converse,
- $(=, \dot{\simeq}_{=_{\mathbb{N}}})$ for $\lhd^{\mathrm{BPullC}}$, and $(\dot{\simeq}_{=_{\mathbb{N}}}, =)$ for its converse, using the trigger $\lhd^3$, and
- $(=, \dot{\simeq}_{=_{\mathbb{N}}})$ for $\lhd^{\mathrm{BPullW}}$, and $(\dot{\simeq}_{=_{\mathbb{N}}}, =)$ for its converse, using the trigger $\lhd^4$.

H.3.7. *Robustness of operational pre-templates and their converse.* For the operational pre-templates and their converse, we use the class $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}\text{-bf}}$ of binding-free contexts. This restriction is crucial to rule out some shallow overlaps.

Using Table 6, shallow overlaps with the operational pre-templates $\lhd^{\vec{@}}$ and $\lhd^{\mathtt{ref}}$ are caused by paths $(\otimes^{\star}_{\mathsf{C}})^{+}$ from the token context. However, the restriction to binding-free contexts makes this situation impossible, which means the robustness check always boils down to Corollary H.8(1) and Corollary H.8(2).

In checking robustness of the converse $(\lhd^{\vec{@}})^{-1}$ and $(\lhd^{\mathtt{ref}})^{-1}$, shallow overlaps are caused by paths:

$$\begin{array}{lll} (\otimes^{\star}_{\mathsf{C}})^{+}, & =, & +, \\ \vec{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}, & := \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}, & -, \\ \mathtt{ref} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}, & !, & -_1 \end{array}$$

from the token target. Like the case of $(\lhd^{\otimes})^{-1}$, all paths but $(\otimes^{\star}_{\mathsf{C}})^{+}$ give rise to a state that is not rooted, which can be checked using Lemma G.2. The paths $(\otimes^{\star}_{\mathsf{C}})^{+}$ are impossible because of the binding-free restriction. As a result, this robustness check also boils down to Corollary H.8(1) and Corollary H.8(2).

H.3.8. *Robustness of the parametricity pre-template* $\lhd^{\mathrm{Param}}$ *and its converse.* These two pre-templates concern lambda-abstractions, and they give rather rare examples of robustness check where we compare different numbers of transitions.

Using Table 6, shallow overlaps with these pre-templates are caused by paths:

$$\begin{array}{ll} (\otimes^{\star}_{\mathsf{C}})^{+}, & \vec{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}, \\ \mathtt{ref} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}, & := \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*} \end{array}$$

from the token target.

As for compute transitions of operations $\mathbb{O}^{\mathrm{ex}}_{\not\iota} \backslash \{\vec{@}\}$, there are two possible situations.

- Shallow overlaps are caused by paths $\mathtt{ref} \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}$ or $:= \cdot (\mathbb{O}^{\mathrm{ex}}_{\checkmark})^{*}$.
- There is no overlap at all, which boils down to Corollary H.8(1).

In the first situation, a stable hypernet G_S of a local rewrite rule (see e.g. Figure 19a) contains shallow edges, labelled with $\lambda \in \mathbb{O}^{\mathrm{ex}}_{\checkmark}$, that are contributed by a pre-template. The overlapped shallow contributions are not modified at all by the rewrite rule, and consequently, one rewrite transition results in a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, =)$.

As for copy transitions, either one path $(\otimes^{\star}_{\mathsf{C}})^{+}$ causes one shallow overlap, or all overlaps are deep. The latter situation boils down to Corollary H.8(1). In the former situation, one lambda-abstraction contributed by a pre-template gets duplicated. Namely, a $\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen with a non-entering rewrite token can be turned into the form $(\mathcal{C}[\dot{\mathcal{C}}'[\chi'], \vec{\chi}]; H^1, \vec{H}^1; H^2, \vec{H}^2)$ where $\dot{\mathcal{C}}'$ is a focussed context in the form of Figure 25b. There exist a focussed context $\dot{\mathcal{C}}''$ and two hypernets $G^1 \lhd^{\mathrm{Param}} G^2$ such that:

$$\mathcal{C}[\dot{\mathcal{C}}'[H^1], \vec{H}^1] \to \mathcal{C}[\dot{\mathcal{C}}''[G^1], \vec{H}^1],$$
$$\mathcal{C}[\dot{\mathcal{C}}'[H^2], \vec{H}^2] \to \mathcal{C}[\dot{\mathcal{C}}''[G^2], \vec{H}^2].$$

Results of these rewrite transitions give a new quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, =)$.

As for compute transitions of the operation '$\overrightarrow{@}$', there are two possible situations.

- One path $\overrightarrow{@}$ causes a shallow overlap of the edge that has label λ and gets eliminated by a micro-beta rewrite rule, and possibly some other paths $\overrightarrow{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\swarrow})^*$ cause shallow overlaps in the stable hypernet G_S (see Figure 10b).
- There are possibly deep overlaps, and paths $\overrightarrow{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\swarrow})^*$ may cause shallow overlaps in the stable hypernet G_S.

In the second situation, all overlaps are not modified at all by the micro-beta rewrite rule, except for some deep overlaps turned shallow. Consequently, one rewrite transition results in a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, =)$.

In the first situation, one lambda-abstraction contributed by the pre-template is modified, while all the other shallow overlaps (if any) are not. We can focus on the lambda-abstraction. The micro-beta rewrite acts on the lambda-abstraction, an edge labelled with '$\overrightarrow{@}$', and the stable hypernet G_S.

The involved lambda-abstraction can be in two forms (see Figure 20). Firstly, it contains function application in its body. Application of the micro-beta rule discloses the inner function application, whose function side is another lambda-abstraction that can be related by the pre-template $\lhd^{\mathrm{Param}}$ again. As a result, one rewrite transition yields a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, =)$. Secondly, the involved lambda-abstraction consists of dereferencing '!' or constant '1'. Application of the micro-beta rule discloses the dereferencing, or constant, edge. When it is the dereferencing edge that is disclosed, the micro-beta rule is followed by a few transitions to perform dereferencing and produce the same constant '1'. As a result, we compare nine transitions with one transition, and obtain a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(\dot{\preceq}_{\geq_{\mathbb{N}}}, =)$, using triggers $\lhd^6$ and $\lhd^9$.

The case of the converse of $\lhd^{\mathrm{Param}}$ is similar. The only difference is that, in the last situation described above where we compare nine transitions with one transition and obtain a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(\dot{\preceq}_{\geq_{\mathbb{N}}}, =)$, we obtain a quasi-$\mathbb{C}_{\mathbb{O}^{\mathrm{ex}}}$-specimen up to $(=, \dot{\preceq}_{\leq_{\mathbb{N}}})$ as a result of the symmetrical comparison of transitions.

On a final note, let us recall Section 12.4, where we observed some situations of robustness for $\lhd^{\mathrm{Param}}$ using informal reduction semantics on terms. We namely observed situations where parts related by the pre-template is subject to the standard call-by-value beta reduction, either as an argument or a function. The involvement as a function corresponds to one of the robustness situations described in this section, namely: a shallow overlap with a micro-beta rewrite rule that is caused by a path $\overrightarrow{@}$ and modified by the rewrite rule. The other involvement, which is as an argument, corresponds to a combination of two situations described in this section, namely: any overlaps with a contraction rule, and shallow overlaps with a micro-beta rule that are caused by paths $\overrightarrow{@} \cdot (\mathbb{O}^{\mathrm{ex}}_{\swarrow})^*$ and preserved. The combination is due to the fact that the universal abstraction machine decomposes the beta reduction into the micro-beta rewrite rule and contraction rules, making substitution explicit and not eager.