Abstract Interpretation of Resolution-Based Semantics

Patrick Cousot

École Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05 (France) & Courant Institute of Math. Sciences, New York University, New York, NY 10012

Radhia Cousot

CNRS & École Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05 (France)

Roberto Giacobazzi

Universitá degli Studi di Verona, Dipartimento di Informatica, Cá Vignal 2 Strada Le Grazie 15, 37134 Verona (Italy)

Abstract

We extend the abstract interpretation point of view on context-free grammars by Cousot and Cousot to resolution-based logic programs and proof systems. Starting from a transition-based small-step operational semantics of PROLOG programs (akin to the Warren Machine), we consider maximal finite derivations for the transition system from most general goals. This semantics is abstracted by instantiation to terms and furthermore to ground terms, following the so called *c* and *s* semantics approach. Orthogonally, these sets of derivations can be abstracted to SLD-trees, call patterns and models, as well as interpreters providing effective implementations (such as PROLOG). These semantics can be presented in bottom-up fixpoint form. This abstract interpretation-based construction leads to classical bottom-up semantics (such as the *s*-semantics of computed answers, the *c*-semantics of correct answers of Keith Clark, and the minimal-model semantics of logical consequences of Maarten van Emden and Robert Kowalski). The approach is general and can be applied to infinite and top-down semantics in a straightforward way.

Key words: Abstract interpretation, Bottom-up semantics, Herbrand semantics, Logic programming, *s*-semantics.

Email addresses: Patrick.Cousot@ens.fr, pcousot@cs.nyu.edu (Patrick Cousot), Radhia.Cousot@ens.fr (Radhia Cousot), roberto.giacobazzi@univr.it (Roberto Giacobazzi).

1 Introduction

The semantics of logic programs is characterised by a variety of forms and methods, ranging from the more traditional operational and denotational semantics towards the more logic-based ones related with the view of the interpreter as a theorem prover. Examples of this variety include the semantics of predicate logic as a programming language in [1,2], the operational and denotational semantics of PROLOG [3–7], the logical models of PROLOG control features [8–10], the (fixpoint) observational models in [11–19], and the so called or-compositional models in [20–25], the last culminating in the so called *s*-semantics approach to logic program semantics [26], which is comprehensive of different observational semantics.

The essence in the study of comparative semantics in logic programming is the attempt to capture the difference between the various aspects of logic as a programming language, ranging from its view in theorem proving to its use in concrete programming. Theorem proving corresponds to restricting programs as theories of definite Horn clauses. In this context both a model-theoretic semantics (unique Herbrand representative model) and a proof-procedure (e.g. SLD resolution) are given [1,27]. Programming instead considers resolution strategies and backtracking control mechanisms such as cut as essential parts of the art of logic programming [28,29]. For these latter aspects, the model-theoretic semantics is not adequate. This wide range of possible interpretation for a logic program has led researchers to develop a number of different semantics capturing specific aspects of logic as a programming language, from operational (resolution-based) semantics to denotational, model-theoretic, etc. Several attempts have been made in order to construct a comprehensive hierarchy of semantics [18], some of them using abstract interpretation for specifying semantics at different levels of abstraction [30–33].

In this paper we develop a hierarchy of semantics for resolution-based languages by incremental abstractions of a maximal trace semantics. The trace semantics is constructed by generalising transitional semantics of context-free grammars akin push-down automata to resolution-based derivations of Horn-like clauses. The result is a hierarchy of top-down and bottom-up semantics of logic programs including as abstract interpretations most of the well-known semantics: the partial correctness semantics, the success semantics, the ground (Herbrand) models, the SLD-semantics, the breadth-first semantics and the cut semantics. All semantics are derived as abstract interpretations, where consecutive abstractions specify a PROLOG interpreter modelling the different observable properties of the program.

2 Mathematical Notations

We let $\mathfrak{B} \triangleq \{true, false\}$ be the *Boolean* truth values ($\wedge$ is conjunction, etc), $i, n, \ldots \in \mathfrak{N}$ be the set of *natural numbers* $\lambda, \mu \in \mathfrak{O}$ be the class of ordinals both with infimum 0 and natural ordering $\leqslant$, $\langle x_i, i \in \Delta \rangle$ be the *indexed family* of elements x_i indexed by $i \in \Delta$ which is a *sequence* when $\langle \Delta, < \rangle$ is totally ordered with infimum (e.g. Δ is $\mathfrak{N}$ or $\mathfrak{O}$). The concatenation of sequences is denoted by juxtaposition and $\langle \langle x_{ij}, j \in \Delta_2 \rangle, i \in \Delta_1 \rangle$ is $\langle x_{ij}, \langle i, j \rangle \in \Delta_1 \times \Delta_2 \rangle$ where $\Delta_1 \times \Delta_2$ is totally ordered lexicographically.

3 Languages

Let $\mathscr{A}$ be an *alphabet*, that is a finite set of *letters*. A *sentence* $\sigma \in \mathscr{A}^\star$ *over the alphabet* $\mathscr{A}$ of length $|\sigma| \triangleq n \geqslant 0$ is a possibly empty finite sequence $\sigma_1\sigma_2 \ldots \sigma_n$ of letters $\sigma_1, \sigma_2, \ldots, \sigma_n \in \mathscr{A}$. For $n = 0$, the empty sentence is denoted ϵ of length $|\epsilon| = 0$. A *language* Σ *over the alphabet* $\mathscr{A}$ is a set of sentences $\Sigma \in \wp(\mathscr{A}^\star)$. We represent concatenation by juxtaposition. It is extended to languages as $\Sigma\Sigma' \triangleq \{\sigma\sigma' \mid \sigma \in \Sigma \wedge \sigma' \in \Sigma'\}$. Given a set $\mathscr{P} \triangleq \{[_i \mid i \in \Delta\} \cup \{]_i \mid i \in \Delta\}$ of matching parentheses and an alphabet $\mathscr{A}$, the *Dyck language* $\mathbb{D}_{\mathscr{P},\mathscr{A}} \subseteq (\mathscr{P} \cup \mathscr{A})^\star$ over $\mathscr{P}$ and $\mathscr{A}$ is the set of well-parenthesized sentences over $\mathscr{P} \cup \mathscr{A}$. In any sentence $\sigma \in \mathbb{D}_{\mathscr{P},\mathscr{A}}$ the number of opening parentheses $[_i$ for $i \in \Delta$ is equal to the number of matching closing parentheses $]_i$ while in any prefix of σ there are no fewer opening parentheses than closing parentheses. A *pure* Dyck language has $\mathscr{A} = \varnothing$. The *parenthesized language* over $\mathscr{P}$ and $\mathscr{A}$ is $\mathbb{P}_{\mathscr{P},\mathscr{A}} \triangleq \{[_i\sigma]_i \mid i \in \Delta \wedge \sigma \in \mathbb{D}_{\mathscr{P},\mathscr{A}} \setminus \{\epsilon\}\}$.

4 Syntax of Logic Programs

We let $\mathbb{f}$ be a set of *function symbols* $f \in \mathbb{f}$, $f/n \in \mathbb{f}/n$ be the subset of function symbols of arity $n \geqslant 0$ (unless otherwise stated $\mathbb{f}/0 \neq \varnothing$), $\mathbb{v}$ be a set of *variable symbols* $v \in \mathbb{v}$ (such that $\mathbb{f} \cap \mathbb{v} = \varnothing$), $\vec{v} \in \vec{\mathbb{v}}$ be possibly empty sequences of variable symbols $\vec{v} = v_1, \ldots, v_n$, $n \geqslant 0$ ($\vec{\epsilon}$ being the empty sequence of variables), $\mathbb{t}$ be the set of *terms* $T, U, \ldots \in \mathbb{t}$ built on $\mathbb{f}$ and $\mathbb{v}$, $\mathbb{p}$ be a set of *predicate symbols* $p \in \mathbb{p}$ (such that $\mathbb{p} \cap \mathbb{v} = \varnothing$ and $\mathbb{p} \cap \mathbb{f} = \varnothing$), $p/n \in \mathbb{p}/n$ be the subset of predicate symbols of arity $n \geqslant 0$, $\mathbb{A}$ be a set of *atoms* $A, B \in \mathbb{A}$ built on $\mathbb{p}$ and $\mathbb{t}$, $\boldsymbol{B} \in \mathbb{B}$ be possibly empty sequences of atoms $\boldsymbol{B} = B_1 \ldots B_n$, $n \geqslant 0$ ($\boldsymbol{\varepsilon}$ being the empty sequence of atoms), $C \in \mathbb{C} \triangleq \mathbb{A} \times \mathbb{B}$ be *definite clauses* of the form $C = A \leftarrow \boldsymbol{B}$ where the *head* $A \in \mathbb{A}$ is an atom and the *body* $\boldsymbol{B} \in \mathbb{B}$ is a sequence of atoms ($\boldsymbol{B}$ is empty for *unit clauses*), $P \in \mathbb{P}^n \triangleq [0, n[\mapsto \mathbb{C}$ be the set of all PROLOG *programs* which are

non-empty sequences of clauses $P = P_0...P_{n-1}$ of length $|P| = n \geqslant 1$, $\mathbb{P} \triangleq \bigcup_{n \geqslant 1} \mathbb{P}^n$ be the set of all PROLOG programs, $\mathbb{L} \triangleq \wp(\mathbb{C}) \setminus \{\varnothing\}$ be the set of *logic programs* $P \in \mathbb{L}$ which are nonempty (unordered) sets of clauses, $\mathbb{G} \triangleq \{p(v) \mid p \in \mathbb{p} \wedge v \in \mathbb{v}\}$ be the set of *most general atomic goals*. There is an obvious abstraction of a PROLOG program $P \in \mathbb{P}^n$ into a logic program $\alpha^{\mathbb{L}}(P) \triangleq \{P_1, \ldots, P_n\} \in \mathbb{L}$ which forgets about the ordering of clauses.

Example 1 The following PROLOG program defines natural numbers ($\mathtt{0} \in \mathbb{f}/0$, $\mathtt{s} \in \mathbb{f}/1$, $\mathtt{n} \in \mathbb{p}/1$ and $x \in \mathbb{v}$).

$$\begin{array}{ll} \mathtt{0}: & \mathtt{n}(\mathtt{0}) \leftarrow \\ \mathtt{1}: & \mathtt{n}(\mathtt{s}(x)) \leftarrow \mathtt{n}(x) \end{array}$$

□

We let $vars(e)$ be the set of variables of the syntactic expression $e \in \mathbb{e}$. If $\mathscr{E} \in \wp(\mathbb{e})$ is a set of syntactic expressions then $ground(\mathscr{E}) \triangleq \{e \in \mathscr{E} \mid vars(e) = \varnothing\}$ is the subset of *ground* expressions. The subset of ground expressions in $\mathbb{e}$ is written $\overline{\mathbb{e}} \triangleq \{e \in \mathbb{e} \mid vars(e) = \varnothing\}$. For example $\overline{\mathbb{t}}$ is the set of all ground terms, $\overline{\mathbb{A}}$ is the set of all ground atoms, etc.

5 Substitutions

A *substitution* $\vartheta, \sigma \in \mathbb{S}$ is a map $\vartheta \in \mathbb{v} \mapsto \mathbb{t}$ whose *domain* $dom(\vartheta) \triangleq \{v \in \mathbb{v} \mid \vartheta(v) \neq v\}$ is finite. The result of applying a substitution ϑ to a term T is the *instance* of T denoted $\vartheta(T)$. We let $inst(T) \triangleq \{\vartheta(T) \mid \vartheta \in \mathbb{S}\}$ be the set of instances of term $T \in \mathbb{t}$ and $inst(\mathscr{T}) \triangleq \bigcup\{\vartheta(T) \mid \vartheta \in \mathbb{S} \wedge T \in \mathscr{T}\}$ be the set of instances of a set $\mathscr{T} \in \wp(\mathbb{t})$ of terms. The *empty substitution* ε has $dom(\varepsilon) = \varnothing$. The *range* of substitution ϑ is $rng(\vartheta) \triangleq \bigcup\{vars(\vartheta(v)) \mid v \in dom(\vartheta)\}$. The *restriction* of a substitution ϑ to the variables $vars(e)$ of a syntactic expression e is $\vartheta|_e$. The *composition* $\vartheta \circ \sigma$ is $\boldsymbol{\lambda}\, v \bullet \vartheta(\sigma(v))$. A substitution ϑ is *idempotent* whenever $\vartheta \circ \vartheta = \vartheta$ or equivalently $dom(\vartheta) \cap rng(\vartheta) = \varnothing$. We let $\mathbb{S}^\circ$ be the set of idempotent substitutions. A *renaming* ρ is a (non-idempotent) substitution which has an *inverse* ρ^{-1} such that $\rho^{-1} \circ \rho = \rho \circ \rho^{-1} = \varepsilon$. The preorder $\preceq$ on substitutions is $\vartheta \preceq \sigma$ (σ "*is more general than*" ϑ) if and only if there exists σ' such that $\vartheta = \sigma' \circ \sigma$. The corresponding equivalence relation is $\vartheta \simeq \vartheta'$ if and only if $\vartheta \preceq \vartheta'$ and $\vartheta' \preceq \vartheta$. $[\vartheta]_\simeq \triangleq \{\vartheta' \in \mathbb{S} \mid \vartheta' \simeq \vartheta\}$ is the equivalence class of $\vartheta \in \mathbb{S}$. $\mathscr{S}/_\simeq \triangleq \{[\vartheta]_\simeq \mid \vartheta \in \mathscr{S}\}$ is the set of equivalence classes of $\mathscr{S} \in \wp(\mathbb{S})$. $\mathbb{S}^\circ/_\simeq$ is the set of idempotent substitutions considered up to renaming. $\langle \mathbb{S}^\circ/_\simeq, \preceq \rangle$ is a complete lattice [34]. It is a complete Heyting algebra when closed by instantiation.

Similarly for terms T and T', $T \preceq T'$ (T' "*is more general than*" T or T "*is an instance of*" T') if and only if there exists a substitution ϑ such that $\vartheta(T') = T$

or equivalently $inst(T) \subseteq inst(T')$. The corresponding equivalence relation is *term renaming* that is $T \simeq T'$ if and only if $T \preceq T'$ and $T' \preceq T$. $\mathscr{T}^{\emptyset}/_{\simeq}$ is the set of equivalence classes $[T]_{\simeq}$, $T \in \mathscr{T}$ augmented with infimum $\emptyset$.

6 Unification

A substitution ϑ is a *unifier* of a set of terms $\mathscr{T} \in \wp(\mathbb{t})$ if and only if $\forall T, T' \in \mathscr{T} : \vartheta(T) = \vartheta(T')$ in which case $\mathscr{T}$ is said to be *unifiable*. A unifiable set of terms $\mathscr{T}$ has an idempotent *most general unifier* σ which is unique up to renaming and we write $mgu(\mathscr{T}) = \{\sigma\}$. By convention, we let $mgu(\mathscr{T}) \triangleq \varnothing$ when $\mathscr{T}$ is not unifiable. This notion of unification with respect to a set of terms is equivalent to unification with respect to a set of equations $E \in \mathbb{E}$ of the form $T = U$ with $T, U \in \mathbb{t}$ where $\mathcal{E} = \{T_i = U_i \mid i \in \Delta\}$ is *unifiable* if there exists a substitution ϑ such that $\forall i \in \Delta : \vartheta(T_i) \simeq \vartheta(U_i)$ in which case there exists a *most general idempotent unifier* $mgu(\mathcal{E})$ *of* $\mathcal{E}$, which is unique up to renaming. The *set of equations corresponding to a substitution* ϑ is $\mathfrak{E}(\vartheta) \triangleq \{v = \vartheta(v) \mid v \in dom(\vartheta)\}$. The *parallel composition* of idempotent substitutions $\uparrow \in \mathbb{S}^{\circ}/_{\simeq} \times \mathbb{S}^{\circ}/_{\simeq} \mapsto \mathbb{S}^{\circ}/_{\simeq}$ is $\vartheta \uparrow \sigma \triangleq mgu(\mathfrak{E}(\vartheta) \cup \mathfrak{E}(\sigma))$ [34], which corresponds to the least upper bound (lub) of classes of idempotent substitutions.

7 Labelled Transition Systems

A *labelled transition system* is a quadruple $\langle \mathscr{E}, \mathscr{L}, \longrightarrow, \mathscr{I} \rangle$ where $\mathscr{E}$ is a non-empty set of *states* η, $\mathscr{L}$ is a non-empty set of *labels* ℓ, $\longrightarrow \in \wp(\mathscr{E} \times \mathscr{L} \times \mathscr{E})$ is the *transition relation* and $\mathscr{I} \subseteq \mathscr{E}$ is the set of *initial states* ι. We write $\eta \xrightarrow{\ell} \eta'$ for $\langle \eta, \ell, \eta' \rangle \in \longrightarrow$ and $\eta \not\longrightarrow$ for $\forall \eta' \in \mathscr{E} : \langle \eta, \ell, \eta' \rangle \notin \longrightarrow$.

8 Traces and maximal derivations

8.1 Finite Traces

A *finite trace* $\theta \in \mathbf{\Theta}[n+1]$ *of length* $|\theta| = n+1$, $n \geqslant 0$, has the form $\theta = \eta_0 \xrightarrow{\ell_0} \eta_1 \ldots \eta_{n-1} \xrightarrow{\ell_{n-1}} \eta_n$ whence it is a pair $\theta = \langle \underline{\theta}, \overline{\theta} \rangle$ where $\underline{\theta} \in [0, n] \mapsto \mathscr{E}$ is a nonempty finite sequence of states $\underline{\theta}_i = \eta_i$, $i = 0, \ldots, n$ and $\overline{\theta} \in [0, n-1] \mapsto \mathscr{L}$ is a finite sequence of labels $\overline{\theta}_j = \ell_j$, $j = 0, \ldots, n-1$ (which is the empty sequence ϵ when $|\theta| = 1$).

A *finite trace* $\theta \in \mathbf{\Theta}^*$ is nonempty, finite, of any length so $\mathbf{\Theta}^* \triangleq \bigcup_{n\in[1,+\infty[} \mathbf{\Theta}[n]$.

The *concatenation* $\theta \xrightarrow{\ell} \theta'$ of traces θ and θ' through label ℓ is extended to sets. We also need the *junction* of sets of traces $\Theta, \Theta' \in \wp(\mathbf{\Theta})$, as follows

$$\Theta \fatsemi \Theta' \triangleq \{\theta \xrightarrow{\ell} \eta \xrightarrow{\ell'} \theta' \mid \theta \xrightarrow{\ell} \eta \in \Theta \wedge \eta' \xrightarrow{\ell'} \theta' \in \Theta' \wedge \eta = \eta'\} \ . \qquad (1)$$

8.2 Maximal Derivations

A *derivation* of the labelled transition system $\mathsf{S} = \langle \mathscr{E}, \mathscr{L}, \longrightarrow, \mathscr{I} \rangle$ is a trace $\theta = \eta_0 \xrightarrow{\ell_0} \eta_1 \ldots \eta_{k-1} \xrightarrow{\ell_{k-1}} \eta_k \ldots$ generated by the transition system S, that is $\forall i \in [0, |\theta|[: \eta_i \xrightarrow{\ell_i} \eta_{i+1}$.

By abuse of notation, a state η is assimilated to the derivation $\theta \in \mathbf{\Theta}[1]$ such that $\underline{\theta}_0 = \eta$ and $\overline{\theta} = \epsilon$, while a transition $\eta \xrightarrow{\ell} \eta'$ is assimilated to the derivation $\theta \in \mathbf{\Theta}[2]$ such that $\underline{\theta}_0 = \eta$, $\overline{\theta}_0 = \ell$ and $\underline{\theta}_1 = \eta'$.

A *prefix derivation* of $\mathsf{S} = \langle \mathscr{E}, \mathscr{L}, \longrightarrow, \mathscr{I} \rangle$ is a derivation of S starting with an initial state $\eta_0 \in \mathscr{I}$. A *suffix derivation* of S is a derivation of S which is finite of length $n = |\theta|$ and ending with an final state $\forall \eta \in \mathscr{E} : \forall \ell \in \mathscr{L} : \neg(\eta_n \xrightarrow{\ell} \eta)$. A *maximal derivation* of the labelled transition system S is both a prefix and a suffix derivation of S.

9 Terminal Labelled Transition System of Prolog Programs

9.1 Labels and Parentheses

We let $\mathscr{L} \triangleq \mathscr{O} \cup \mathscr{C}$ be the set of *labels* $\ell \in \mathscr{L}$ where $\mathscr{O} \triangleq \{(\!|\, \mathtt{i}{:}C/\sigma \mid \ \mathtt{i} \in \mathfrak{N} \wedge C \in \mathbb{C} \wedge \sigma \in \mathbb{S}\}$ is the set of *opening parentheses* while $\mathscr{C} \triangleq \{\mathtt{i}{:}C \,|\!)\} \mid C \in \mathbb{C} \wedge \mathtt{i} \in \mathfrak{N}\}$ is the set of *closing parentheses*. A matching pair of parentheses $(\!|\, \mathtt{i}{:}C/\sigma \ldots \mathtt{i}{:}C \,|\!)$ delimits a derivation for the labelled clause $\mathtt{i}{:}C$ instantiated by substitution σ.

9.2 Stacks

In the following we use the grammar LALR-based notation in [35] for sets of clauses. We let *stacks* $\varpi \in \mathscr{S} \triangleq \mathscr{K}^+$ for a program $P \in \mathbb{P}$ be non-empty

sequences of *control states* $\kappa \in \mathscr{K} \triangleq \mathbb{C}^\bullet \cup \mathscr{M}$ which are either a *clause state* in $\mathbb{C}^\bullet \triangleq \{[\mathtt{i}\!:\!A \leftarrow \boldsymbol{B}.\boldsymbol{B}'] \mid \mathtt{i}\!:\!A \leftarrow \boldsymbol{B}\,\boldsymbol{B}' \in P\}$ specifying the control state of the derivation ($\boldsymbol{B}$ has been derived while $\boldsymbol{B}'$ is still to be derived) or a *marker* $\mathscr{M} = \{[\vdash A], [\dashv\!\Box] \mid A \in \mathbb{A}\}$ where $[\vdash A]$ is the initial stack marker while $[\dashv\!\Box]$ is the final empty stack marker for the beginning (resp. the end) of a derivation for the initial question $A \in \mathbb{A}$. The *height* of a stack ϖ is its length $|\varpi|$.

9.3 States

We let *states* $\eta \in \mathscr{E} \triangleq \mathscr{S} \times \mathbb{S}$ be pairs $\eta = \langle \varpi, \vartheta \rangle$ of a stack ϖ and a substitution ϑ. The stack ϖ specifies a return point, i.e., the corresponding clauses, after a procedure call for a clause while the substitution ϑ is returned by the call.

9.4 Prolog *Labelled Transition System*

Given a Prolog program $P \in \mathbb{P}$, we define a *concrete labelled transition system* $\mathsf{S}^{\mathsf{t}}[\![P]\!] \triangleq \langle \mathscr{E}, \mathscr{L}, \longrightarrow^{\mathsf{t}}, \mathscr{I} \rangle$ (akin to the Warren machine [36,37]). The set of initial states is $\mathscr{I} \triangleq \{\langle [\vdash A], \vartheta \rangle \mid A \in \mathbb{A} \wedge \vartheta \in \mathbb{S}\}$ where $\langle [\vdash A], \vartheta \rangle$ specifies the *goal* $\vartheta(A)$ (most often ϑ is chosen as the empty substitution ε) [1]. Let $\mathtt{i}\!:\!A \leftarrow \boldsymbol{B} \Subset P$ means that $\mathtt{i}\!:\!A \leftarrow \boldsymbol{B}$ is a clause of the Prolog program P renamed/standardized apart using fresh variables [38]. The labelled transition relation $\xrightarrow{\ell}{}^{\mathsf{t}}$, $\ell \in \mathscr{L}$ is

$$\langle [\vdash A], \vartheta \rangle \xrightarrow{(\!|\, \mathtt{i}:A' \leftarrow \boldsymbol{B}/\sigma}{}^{\mathsf{t}} \langle [\dashv\!\Box][\mathtt{i}\!:\!A' \leftarrow \boldsymbol{.B}], \vartheta' \rangle$$
$$\text{if} \quad \mathtt{i}\!:\!A' \leftarrow \boldsymbol{B} \Subset P,\ \sigma \in mgu(\vartheta(A), A'),\ \vartheta' \in \sigma \uparrow \vartheta \tag{2}$$

$$\langle \varpi[\mathtt{i}\!:\!A \leftarrow \boldsymbol{B}.B\boldsymbol{B}'], \vartheta \rangle \xrightarrow{(\!|\, \mathtt{j}:B' \leftarrow \boldsymbol{B}''/\sigma}{}^{\mathsf{t}} \langle \varpi[\mathtt{i}\!:\!A \leftarrow \boldsymbol{B}B.\boldsymbol{B}'][\mathtt{j}\!:\!B' \leftarrow \boldsymbol{.B}''], \vartheta' \rangle$$
$$\text{if} \quad \mathtt{i}\!:\!A \leftarrow \boldsymbol{B}B\boldsymbol{B}', \mathtt{j}\!:\!B' \leftarrow \boldsymbol{B}'' \Subset P,\ \sigma \in mgu(\vartheta(B), B'),\ \vartheta' \in \sigma \uparrow \vartheta \tag{3}$$

$$\langle \varpi[\mathtt{i}\!:\!A \leftarrow \boldsymbol{B}.], \vartheta \rangle \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}\,|\!)}{}^{\mathsf{t}} \langle \varpi, \vartheta \rangle \qquad \text{if} \quad \mathtt{i}\!:\!A \leftarrow \boldsymbol{B} \Subset P\ . \tag{4}$$

Examples of transitions $\xrightarrow{\ell}{}^{\mathsf{t}}$ are given in **Ex. 2** below.

The intuition of (2) is that the goal $\vartheta(A)$ is unified with the head A' of the renamed apart clause $\mathtt{i}\!:\!A' \leftarrow \boldsymbol{B}$ of the Prolog program by the most general substitution σ and so it remains to prove $\sigma \uparrow \vartheta(\boldsymbol{B})$ so $[\mathtt{i}\!:\!A' \leftarrow \boldsymbol{.B}]$ is pushed on the stack and $\sigma \uparrow \vartheta$ is recorded in the state. In particular for the empty

[1] A conjunction of goals can be handled by adding a clause to the program.

substitution $\langle[\vdash A],\, \varepsilon\rangle \xrightarrow{(\!|\, \mathtt{i}:A'\leftarrow \boldsymbol{B}/\sigma}{}^{\mathsf{t}} \langle[\dashv\Box][\mathtt{i}:A' \leftarrow \boldsymbol{.B}],\, \sigma\rangle$ and it remains to prove $\sigma(\boldsymbol{B})$.

If and when the proof succeeds, the final marker $[\dashv\Box]$ on the stack will indicate that the proof is finished while the substitution ϑ'' in the final state $\langle[\dashv\Box],\, \vartheta''\rangle$ will be the *answer substitution*.

The intuition of (3) is that the subgoal $\vartheta(B)$ is unified with the head B' of the renamed apart clause $\mathtt{j}:B' \leftarrow \boldsymbol{B}''$ of the PROLOG program by the most general substitution σ and so it remains first to prove $\sigma \uparrow \vartheta(\boldsymbol{B}'')$ so $[\mathtt{j}:B' \leftarrow \boldsymbol{.B}'']$ is pushed on the stack and $\sigma \uparrow \vartheta$ is recorded in the state, and second to prove $\boldsymbol{B}'$ as indicated by the control state $[\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{.B}']$ on the stack and finally to terminate the proof as indicated by the bottom ϖ of the stack.

The intuition of (4) is that the proof of $\boldsymbol{B}$ is finished and so the proof goes on as indicated by the bottom ϖ of the stack.

The final states are either

- *answer substitution* states in $\mathscr{E}^{\text{AS}} \triangleq \{\langle[\dashv\Box],\, \vartheta\rangle \mid \vartheta \in \mathbb{S}\}$ for *successful traces*, or
- *finite failure* states in $\mathscr{E}^{\text{FF}} \triangleq \{\langle\varpi[\mathtt{i}:A \leftarrow \boldsymbol{.}B\boldsymbol{B}'],\, \vartheta\rangle \mid \forall \mathtt{j}:B' \leftarrow \boldsymbol{B}'' \notin P : \mathit{mgu}(\vartheta(B), B') = \varnothing\}$ for *failing traces*.

10 Most General Maximal Terminal Derivation Semantics of Logic Programs

10.1 Maximal Derivations of Logic Programs

The maximal derivations of a PROLOG program $P \in \mathbb{P}$ are traces for the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!] \triangleq \langle\mathscr{E},\, \mathscr{L},\, \longrightarrow^{\mathsf{t}},\, \mathscr{I}\rangle$, as defined in **Sec. 8.2**.

Example 2 A maximal derivation for the ground goal $\mathtt{n}(\mathtt{s}(\mathtt{s}(0)))$ (the encoding of the natural number 2) as defined by the PROLOG program of **Ex. 1** is:

$\langle[\vdash \mathtt{n}(\mathtt{s}(\mathtt{s}(0)))],\, \varepsilon\rangle$ ⟅initial state⟆

$\xrightarrow{(\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x))\leftarrow\mathtt{n}(x)/\{x\leftarrow\mathtt{s}(0)\}}{}^{\mathsf{t}}$ ⟅by (2)⟆

$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x)) \leftarrow \boldsymbol{.}\mathtt{n}(x)],\, \{x \leftarrow \mathtt{s}(0)\}\rangle$

$\xrightarrow{(\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')/\{x'\leftarrow 0\}}{}^{\mathsf{t}}$ ⟅by (3)⟆

$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x))\leftarrow\mathtt{n}(x)\centerdot][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\centerdot\mathtt{n}(x')],\ \{x\leftarrow\mathtt{s}(\mathtt{0}),x'\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\langle\!|\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow/\varepsilon}_{\mathtt{t}} \qquad \text{⟅by (3)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x))\leftarrow\mathtt{n}(x)\centerdot][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\centerdot][\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow\centerdot],$$
$$\{x\leftarrow\mathtt{s}(\mathtt{0}),x'\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow|\!\rangle}_{\mathtt{t}} \qquad \text{⟅by (4)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x))\leftarrow\mathtt{n}(x)\centerdot][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\centerdot],\ \{x\leftarrow\mathtt{s}(\mathtt{0}),x'\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')|\!\rangle}_{\mathtt{t}} \qquad \text{⟅by (4)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x))\leftarrow\mathtt{n}(x)\centerdot],\ \{x\leftarrow\mathtt{s}(\mathtt{0}),x'\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x))\leftarrow\mathtt{n}(x)|\!\rangle}_{\mathtt{t}} \qquad \text{⟅by (4)⟆}$$
$$\langle[\dashv\Box],\ \{x\leftarrow\mathtt{s}(\mathtt{0}),x'\leftarrow\mathtt{0}\}\rangle \qquad \Box$$

Example 3 A maximal derivation for the most general non-ground goal $\mathtt{n}(x)$ as defined by the PROLOG program of **Ex. 1** is (among many others):

$$\langle[\vdash\mathtt{n}(x)],\ \varepsilon\rangle \qquad \text{⟅initial state⟆}$$
$$\xrightarrow{\langle\!|\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')/\{x\leftarrow\mathtt{s}(x')\}}_{\mathtt{t}} \qquad \text{⟅by (2)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\centerdot\mathtt{n}(x')],\ \{x\leftarrow\mathtt{s}(x')\}\rangle$$
$$\xrightarrow{\langle\!|\mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow\mathtt{n}(x'')/\{x'\leftarrow\mathtt{s}(x'')\}}_{\mathtt{t}} \qquad \text{⟅by (3)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\centerdot][\mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow\centerdot\mathtt{n}(x'')],$$
$$\{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{s}(x'')\}\rangle$$
$$\xrightarrow{\langle\!|\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow/\{x''\leftarrow\mathtt{0}\}}_{\mathtt{t}} \qquad \text{⟅by (3)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\centerdot][\mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow\mathtt{n}(x'')\centerdot][\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow\centerdot],$$
$$\{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{s}(x''),x''\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow|\!\rangle}_{\mathtt{t}} \qquad \text{⟅by (4)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\centerdot][\mathtt{1}:\mathtt{n}(x'')\leftarrow\mathtt{n}(x'')\centerdot],$$
$$\{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{s}(x''),x''\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow\mathtt{n}(x'')|\!\rangle}_{\mathtt{t}} \qquad \text{⟅by (4)⟆}$$
$$\langle[\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\centerdot],\ \{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{s}(x''),x''\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')|\!\rangle}_{\mathtt{t}} \qquad \text{⟅by (4)⟆}$$
$$\langle[\dashv\Box],\ \{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{s}(x''),x''\leftarrow\mathtt{0}\}\rangle \qquad \Box$$

The *selection* of the traces in a set $\Theta \in \wp(\mathbf{\Theta})$ of traces for an atom $A \in \mathscr{A}$ is denoted $\Theta.A$ and defined as

$$\Theta.A \triangleq \{\eta \xrightarrow{(\!|\, \mathtt{i}:A' \leftarrow \boldsymbol{B}/\sigma} \theta \mid \eta \xrightarrow{(\!|\, \mathtt{i}:A' \leftarrow \boldsymbol{B}/\sigma} \theta \notin \Theta \wedge A \simeq A' \wedge \eta \in \mathscr{E} \wedge \theta \in \mathbf{\Theta}\} \quad (5)$$

and similarly the traces starting with a given state $\eta \in E$ are denoted $\Theta.\eta$ defined as

$$\Theta.\eta \triangleq \{\eta' \xrightarrow{(\!|\, \mathtt{i}:A \leftarrow \boldsymbol{B}/\sigma} \theta \mid \eta \simeq \eta' \wedge \eta' \xrightarrow{(\!|\, \mathtt{i}:A \leftarrow \boldsymbol{B}/\sigma} \theta \notin \Theta\} \; . \quad (6)$$

10.2 Transitional Most General Maximal Derivation Semantics

The *most general maximal derivation semantics* $\mathsf{S}^{\mathsf{d}}[\![P]\!] \in \wp(\mathbf{\Theta})$ of a PROLOG program $P \in \mathbb{P}$ is the set of all possible maximal derivations for the concrete labelled transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ of this program P (defined by (2)—(4)) starting from most general goals $\{p(v) \mid p \in \mathbb{p} \wedge v \in \mathbb{v}\}$.

$$\begin{aligned}\mathsf{S}^{\mathsf{d}}[\![P]\!] \triangleq \{&\eta_0 \xrightarrow{\ell_0} \eta_1 \ldots \eta_{n-1} \xrightarrow{\ell_{n-1}} \eta_n \in \mathbf{\Theta}[n+1] \mid n \geqslant 0 \wedge \\ &\eta_0 = \langle [\vdash p(v)],\, \varepsilon\rangle \wedge p \in \mathbb{p} \wedge v \in \mathbb{v} \wedge \forall i \in [0, n-1] : \eta_i \xrightarrow{\ell_i}{}^{\mathsf{t}} \eta_{i+1} \wedge \\ &\forall \eta \in \mathscr{S} : \forall \ell \in \mathscr{L} : \neg(\eta_n \xrightarrow{\ell}{}^{\mathsf{t}} \eta)\} \; .\end{aligned} \quad (7)$$

By def. (2)—(4) of $\longrightarrow^{\mathsf{t}}$, a final state η_f such that $\forall \eta \in \mathscr{S} : \forall \ell \in \mathscr{L} : \neg(\eta_f \xrightarrow{\ell}{}^{\mathsf{t}} \eta)$ is an answer substitution state $\eta_f \in \mathscr{E}^{\text{AS}}$ (of the form $\eta_f = \langle [\dashv\!\Box], \vartheta\rangle$ where ϑ is the computed answer) or is a finite failure state $\eta_f \in \mathscr{E}^{\text{FF}}$.

Example 4 The trace for $\mathtt{n}(x)$ for the PROLOG program of **Ex. 1** given in the **Ex. 3** is a most general maximal derivation while the trace for $\mathtt{n}(\mathtt{s}(\mathtt{s}(\mathtt{0})))$ given in the **Ex. 2** is not. □

Semantic derivations are well-parenthesized so that the structure of computations can be described by trees. Let us define the *parenthesis abstraction* α^p as follows

$$\begin{aligned}\alpha^p(\varpi\varpi') &\triangleq \alpha^p(\varpi')\alpha^p(\varpi), && \text{for stacks}\\ \alpha^p([\vdash A]) &\triangleq \epsilon\\ \alpha^p([\dashv\!\Box]) &\triangleq \epsilon\\ \alpha^p([\mathtt{i}:A \leftarrow \boldsymbol{B}.\boldsymbol{B}']) &\triangleq \mathtt{i}:A \leftarrow \boldsymbol{B}\boldsymbol{B}' \,|\!)\\ \alpha^p((\!|\, \mathtt{i}:A \leftarrow \boldsymbol{B}/\sigma) &\triangleq (\!|\, \mathtt{i}:A \leftarrow \boldsymbol{B}, && \text{for labels}\end{aligned}$$

$$\begin{aligned}
\alpha^p(\mathtt{i}:A \leftarrow \boldsymbol{B}\,\rangle) &\triangleq \mathtt{i}:A \leftarrow \boldsymbol{B}\,\rangle \\
\alpha^p(\langle \varpi,\, \vartheta\rangle) &\triangleq \alpha^p(\varpi), && \text{for states} \\
\alpha^p(\eta_0 \xrightarrow{\ell_0} \eta_1 \ldots \eta_{n-1} \xrightarrow{\ell_{n-1}} \eta_n) &\triangleq \alpha^p(\ell_0)\alpha^p(\ell_1)\ldots\alpha^p(\ell_{n-1})\alpha^p(\eta_n) && \text{for traces.}
\end{aligned}$$

Example 5 The parenthesis abstraction of the following prefix of the maximal derivation given in **Ex. 3** for the PROLOG program of **Ex. 1** and the non-ground goal $\mathtt{n}(x)$

$$\begin{array}{l}
\quad \langle [\vdash \mathtt{n}(x)],\, \varepsilon\rangle \\
\xrightarrow{\langle\!\langle\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')/\{x\leftarrow\mathtt{s}(x')\}}{}_{\mathtt{t}} \\
\quad \langle [\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \boldsymbol{.}\mathtt{n}(x')],\, \{x \leftarrow \mathtt{s}(x')\}\rangle \\
\xrightarrow{\langle\!\langle\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow\mathtt{n}(x'')/\{x'\leftarrow\mathtt{s}(x'')\}}{}_{\mathtt{t}} \\
\quad \langle [\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x')\boldsymbol{.}][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'')) \leftarrow \boldsymbol{.}\mathtt{n}(x'')],\, \{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{s}(x'')\}\rangle
\end{array}$$

is

$$\begin{array}{l}
\langle\!\langle\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x')\ \langle\!\langle\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x'')) \leftarrow \mathtt{n}(x'')\ \mathtt{1}:\mathtt{n}(\mathtt{s}(x'')) \leftarrow \mathtt{n}(x'')\,\rangle\!\rangle \\
\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x')\,\rangle\!\rangle\ .
\end{array}$$

□

Lemma 6 *For any prefix derivation θ of a program P, $\alpha^p(\theta) \in \mathbb{D}_{\mathscr{P},\varnothing}$ is a pure Dyck language.* □

PROOF Let

$$\theta = \langle [\vdash A],\, \varepsilon\rangle \xrightarrow{\ell_0} \langle \varpi_1,\, \vartheta_1\rangle \ldots \langle \varpi_{n-1},\, \vartheta_{n-1}\rangle \xrightarrow{\ell_{n-1}} \langle \varpi_n,\, \vartheta_n\rangle \in \mathsf{S}^{\mathsf{d}}[\![P]\!]\ .$$

The proof is by induction on the length of θ so that we assume, by induction hypothesis, that $\alpha^p(\langle [\vdash A],\, \varepsilon\rangle \xrightarrow{\ell_0} \langle \varpi_1,\, \vartheta_1\rangle \ldots \langle \varpi_{n-2},\, \vartheta_{n-2}\rangle \xrightarrow{\ell_{n-2}} \langle \varpi_{n-1},\, \vartheta_{n-1}\rangle)$ is well-parenthesized.

By definition of $\mathsf{S}^{\mathsf{d}}[\![P]\!]$, we have that $\varpi_{n-1} \neq [\dashv\Box]$ so that ϖ_{n-1} has the form $\varpi_{n-1} = [\vdash A]$ or $\varpi_{n-1} = \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}\boldsymbol{.}\boldsymbol{B}']$.

— In case 1, $\varpi_{n-1} = [\vdash A]$, we have $n = 1$ by definition of $\xrightarrow{\ell}{}^{\mathtt{t}}$ and so

$$\langle \varpi_{n-1},\, \vartheta_{n-1}\rangle \xrightarrow{\ell_{n-1}} \langle \varpi_n,\, \vartheta_n\rangle = \langle [\vdash A],\, \varepsilon\rangle \xrightarrow{\langle\!\langle\, \mathtt{i}:A'\leftarrow\boldsymbol{B}/\sigma} \langle [\dashv\Box][\mathtt{i}:A' \leftarrow \boldsymbol{.}\boldsymbol{B}],\, \sigma\rangle$$

by (2) where $\mathtt{i}:A' \leftarrow \boldsymbol{B} \in P$ and $\sigma \in mgu(A, A')$. By definition of $\mathsf{S}^{\mathsf{d}}[\![P]\!]$, we have

$$\begin{aligned}\alpha^p(\theta) &= \alpha^p(\langle[\vdash A],\, \varepsilon\rangle \xrightarrow{(\!|\, \mathtt{i}:'\leftarrow \boldsymbol{B}/\sigma} \langle[\dashv\Box][\mathtt{i}:A' \leftarrow \boldsymbol{.B}],\, \sigma\rangle)\\ &= (\!|\, \mathtt{i}:A' \leftarrow \boldsymbol{B}/\sigma\ \mathtt{i}:A' \leftarrow \boldsymbol{B}/\sigma \,|\!)\end{aligned}$$

which is well-parenthesized.

— In case 2, $\varpi_{n-1} = \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B.}B\boldsymbol{B}']$,

$$\langle\varpi_{n-1},\, \vartheta_{n-1}\rangle \xrightarrow{\ell_{n-1}} \langle\varpi_n,\, \vartheta_n\rangle =$$
$$\langle\varpi[\mathtt{i}:A \leftarrow \boldsymbol{B.}B\boldsymbol{B}'],\, \vartheta\rangle \xrightarrow{(\!|\, \mathtt{j}:B'\leftarrow \boldsymbol{B}''/\sigma}{}^{\mathtt{t}}\ \langle\varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{.B}'][\mathtt{j}:B' \leftarrow \boldsymbol{.B}''],\, \vartheta'\rangle$$

by (3) where $\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{B}', \mathtt{j}:B' \leftarrow \boldsymbol{B}'' \in P$, $\sigma \in mgu(\vartheta(B), B')$, $\vartheta' \in \sigma \uparrow \vartheta$. So $\alpha^p(\theta) = \alpha^p(\ell_0)\ldots(\!|\, \mathtt{j}:B' \leftarrow \boldsymbol{B}''/\sigma)\alpha^p([\mathtt{j}:B' \leftarrow \boldsymbol{.B}''])\alpha^p([\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{.B}'])\alpha^p(\varpi) = \alpha^p(\ell_0)\ldots(\!|\, \mathtt{j}:B' \leftarrow \boldsymbol{B}''/\sigma\ [\mathtt{j}:B' \leftarrow \boldsymbol{.B}'']\,|\!)\ [\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{.B}']\,|\!)\ \alpha^p(\varpi)$ which is well-parenthesized if and only if $\alpha^p(\ell_0)\ldots[\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{.B}']\,|\!)\ \alpha^p(\varpi)$ is well-parenthesized, which is the case if and only if $\alpha^p(\langle[\vdash A],\, \varepsilon\rangle \xrightarrow{\ell_0} \langle\varpi_1,\, \vartheta_1\rangle\ldots\langle\varpi_{n-1},\, \vartheta_{n-1}\rangle)$ is well-parenthesized, which is true by induction hypothesis.

— In case 3, $\varpi_{n-1} = \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B.}]$,

$$\langle\varpi_{n-1},\, \vartheta_{n-1}\rangle \xrightarrow{\ell_{n-1}} \langle\varpi_n,\, \vartheta_n\rangle = \langle\varpi[\mathtt{i}:A \leftarrow \boldsymbol{B.}],\, \vartheta\rangle \xrightarrow{\mathtt{i}:A\leftarrow\boldsymbol{B}\,|\!)}{}^{\mathtt{t}}\ \langle\varpi,\, \vartheta\rangle$$

by (4) where $\mathtt{i}:A \leftarrow \boldsymbol{B} \in P$. In this case, we have

$$\begin{aligned}\alpha^p(\theta) &= \alpha^p(\ell_0)\ldots\alpha^p(\ell_{n-1})\ \mathtt{i}:A \leftarrow \boldsymbol{B}\,|\!)\ \alpha^p(\varpi)\\ &= \alpha^p(\ell_0)\ldots\alpha^p(\ell_{n-1})\ \alpha^p(\varpi[\mathtt{i}:A \leftarrow \boldsymbol{B.}])\\ &= \alpha^p(\langle[\vdash A],\, \varepsilon\rangle \xrightarrow{\ell_0} \langle\varpi_1,\, \vartheta_1\rangle\ldots\langle\varpi_{n-1},\, \vartheta_{n-1}\rangle)\end{aligned}$$

which is a pure Dyck language by induction hypothesis. ∎

In particular, **Lem. 6** implies that a maximal succesfull derivation $\theta = \eta_0 \xrightarrow{\ell_0} \eta_1\ldots\eta_{n-1} \xrightarrow{\ell_{n-1}} \eta_n$, $\eta_n \in \mathscr{E}^{\mathrm{AS}}$ of P is well-parenthesized in that $\alpha^p(\theta) = \alpha^p(\ell_0)\alpha^p(\ell_1)\ldots\alpha^p(\ell_{n-1}) \in \mathbb{D}_{\mathscr{P},\varnothing}$ is a pure Dyck language.

11 The Hierarchy of Abstractions

We define abstractions of sets of most general derivations to get classical semantics of PROLOG and logic programs.

11.1 The Partial Correctness Abstractions

The derivations in the most general maximal derivations semantics $\mathsf{S}^{\mathsf{d}}[\![P]\!]$ have finite success and finite failure derivations. The *partial correctness abstractions* forget about finite failures.

11.1.1 Success Abstraction

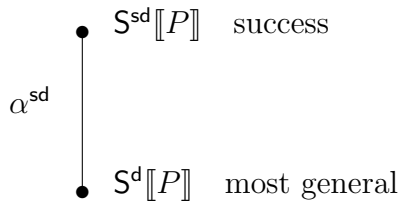
The *success abstraction* eliminates finite failures

$$\alpha^{\mathsf{sd}}(\Theta) \triangleq \{\theta \xrightarrow{\ell} \langle[\dashv\Box],\, \vartheta\rangle \mid \vartheta \in \mathbb{S} \wedge \theta \xrightarrow{\ell} \langle[\dashv\Box],\, \vartheta\rangle \in \Theta\}$$

Note that the instantiation of a failure (i.e., a failing derivation) is still a failure so no potential success behavior is eliminated but the instantiation of a potential finite success behavior might be a finite failure so not all instantiated finite failures might have been eliminated yet (see e.g. **Sec. 11.2.1**).

11.1.2 The Partial Correctness Abstraction Hierarchy

Defining the *partial correctness semantics* $\mathsf{S}^{\mathsf{sd}}[\![P]\!] \triangleq \alpha^{\mathsf{sd}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])$, we get the first dimension in our hierarchy of semantics:



11.2 The Derivation Instantiation Abstractions

The most general maximal derivation semantics $\mathsf{S}^{\mathsf{d}}[\![P]\!]$ for most general goals $[\vdash p(v)]$, $p \in \mathbb{p}$, $v \in \mathbb{v}$ can be abstracted by instantiating the derivations by non-ground or ground substitutions.

11.2.1 The Derivation Non-Ground Instantiation Abstraction

The *derivation instantiation abstraction* maps derivations for most general goals to derivations for instantiations of these goals.

$$\alpha'^{\mathsf{id}}(\langle \varpi,\, \vartheta\rangle)\sigma \triangleq \langle\langle \varpi,\, \vartheta'\rangle,\, b\rangle \qquad \text{where} \quad b = (\vartheta' \in \vartheta \uparrow \sigma)$$

$$\alpha^{\mathsf{id}}(\langle [\vdash p(v)],\, \varepsilon\rangle)\sigma \triangleq \langle [\vdash \sigma(p(v))],\, \sigma\rangle$$

let $\langle \eta_2',\, b\rangle = \alpha'^{\mathsf{id}}(\eta_2)\sigma$ in

$$\begin{aligned}
\alpha^{\mathsf{id}}(\eta_1 \xrightarrow{(\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta \,|\!)^{\mathsf{t}}} \eta_2 \xrightarrow{\ell} \theta)\sigma &\triangleq \eta_1 \xrightarrow{(\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta' \,|\!)^{\mathsf{t}}} \alpha^{\mathsf{id}}(\eta_2' \xrightarrow{\ell} \theta)\sigma && \text{if } b \wedge \vartheta' \in \vartheta \uparrow \sigma \\
&\triangleq \eta_1 && \text{if } \neg b \vee \vartheta' \notin \vartheta \uparrow \sigma \\
\alpha^{\mathsf{id}}(\eta_1 \xrightarrow{\mathtt{i}:A\leftarrow \boldsymbol{B}\,|\!)^{\mathsf{t}}} \eta_2 \xrightarrow{\ell} \theta)\sigma &\triangleq \eta_1 \xrightarrow{\mathtt{i}:A\leftarrow \boldsymbol{B}\,|\!)^{\mathsf{t}}} \alpha^{\mathsf{id}}(\eta_2' \xrightarrow{\ell} \theta)\sigma && \text{if } b \\
&\triangleq \eta_1 && \text{if } \neg b \\
\alpha^{\mathsf{id}}(\eta_1 \xrightarrow{\mathtt{i}:A\leftarrow \boldsymbol{B}\,|\!)^{\mathsf{t}}} \langle [\dashv\!\square],\, \vartheta\rangle)\sigma &\triangleq \eta_1 \xrightarrow{\mathtt{i}:A\leftarrow \boldsymbol{B}\,|\!)^{\mathsf{t}}} \langle [\dashv\!\square],\, \vartheta'\rangle && \text{if } \vartheta' \in \vartheta \uparrow \sigma \\
&\triangleq \eta_1 && \text{if } \vartheta' \notin \vartheta \uparrow \sigma
\end{aligned}$$

$$\alpha^{\mathsf{id}}(\Theta) \triangleq \{\alpha^{\mathsf{id}}(\theta)\sigma \mid \theta \in \Theta \wedge \sigma \in \mathbb{S}\}$$

The initial substitution is propagated along traces unless some instantiation fails along the trace, in which case the trace is truncated, now finishing in a finite failure.

Example 7 The PROLOG program

$$\begin{aligned}
&\mathtt{0}: && \mathtt{n}(\mathtt{0}) \leftarrow \\
&\mathtt{1}: && \mathtt{n}(\mathtt{s}(x)) \leftarrow \mathtt{n}(x) \\
&\mathtt{2}: && \mathtt{p}(\mathtt{a}) \leftarrow
\end{aligned}$$

has the following most general derivation

$$\langle [\vdash \mathtt{n}(x)],\, \varepsilon\rangle$$

$$\xrightarrow{(\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow \mathtt{n}(x')/\{x\leftarrow \mathtt{s}(x')\} \,|\!)^{\mathsf{t}}}$$

$$\langle [\dashv\!\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \centerdot\mathtt{n}(x')],\, \{x \leftarrow \mathtt{s}(x')\}\rangle$$

$$\xrightarrow{(\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow \mathtt{n}(x'')/\{x'\leftarrow \mathtt{s}(x'')\} \,|\!)^{\mathsf{t}}}$$

$$\langle [\dashv\!\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x')\centerdot][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'')) \leftarrow \centerdot\mathtt{n}(x'')],$$

$$\{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{s}(x'')\}\rangle$$

$$\xrightarrow{(\!|\, \mathtt{1}:\mathtt{n}(\mathtt{0})\leftarrow/\{x''\leftarrow \mathtt{0}\} \,|\!)^{\mathsf{t}}}$$

$$\langle [\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x').][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'')) \leftarrow \mathtt{n}(x'').][\mathtt{1}:\mathtt{n}(\mathtt{0}) \leftarrow .],$$
$$\{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{s}(x''), x'' \leftarrow \mathtt{0}\}\rangle$$

$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{0})\leftarrow \rangle\!\rangle}_{\mathtt{t}}$$

$$\langle [\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x').][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'')) \leftarrow \mathtt{n}(x'').],$$
$$\{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{s}(x''), x'' \leftarrow \mathtt{0}\}\rangle$$

$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x''))\leftarrow\mathtt{n}(x'')\rangle\!\rangle}_{\mathtt{t}}$$

$$\langle [\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow \mathtt{n}(x').],\ \{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{s}(x''), x'' \leftarrow \mathtt{0}\}\rangle$$

$$\xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\rangle\!\rangle}_{\mathtt{t}}$$

$$\langle [\dashv\Box],\ \{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{s}(x''), x'' \leftarrow \mathtt{0}\}\rangle$$

The instance for the substitution $\{x \leftarrow \mathtt{s}(\mathtt{a})\}$ leads to the following finite failure

$$\langle [\vdash \mathtt{n}(\mathtt{s}(\mathtt{a}))],\ \{x \leftarrow \mathtt{s}(\mathtt{a})\}\rangle$$

$$\xrightarrow{\langle\!\langle \mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')/\{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{a}\}}_{\mathtt{t}}$$

$$\langle [\dashv\Box][\mathtt{1}:\mathtt{n}(\mathtt{s}(x')) \leftarrow .\mathtt{n}(x')],\ \{x \leftarrow \mathtt{s}(x'), x' \leftarrow \mathtt{a}\}\rangle$$

since $\{x' \leftarrow \mathtt{a}\} \uparrow \{x' \leftarrow \mathtt{s}(x'')\} = \varnothing$. □

More generally, the instantiation of a finite success or finite failure can lead to an earlier finite failure.

11.2.2 The Derivation Ground Instantiation Abstraction

The *derivation ground instantiation abstraction* maps derivations for non-ground goals to derivations for ground instantiations of these goals. The initial ground substitution $\sigma \in \overline{\mathbb{S}}$ is propagated along traces unless the instantiation fails in which case the trace is ignored.

$$\alpha^{\mathsf{gd}}(\Theta) \triangleq \{\alpha^{\mathsf{id}}(\theta)\sigma \mid \theta \in \Theta \wedge \sigma \in \overline{\mathbb{S}}\}$$

Since program clauses are replaced by their ground instantiations, it is no longer necessary to keep track of substitutions [2].

[2] In the following we use the above definition of ground derivations with (useless) substitutions so as not to have to consider the particular case where these substitutions are dropped. So non-ground and ground derivations can be handled uniformely.

$$\alpha'^{\mathsf{gd}}(\langle \varpi,\, \vartheta \rangle)\sigma \triangleq \langle \varpi,\, b \rangle \qquad \text{where} \quad b = (\vartheta' \in \vartheta \uparrow \sigma)$$

let $\langle \varpi'_2,\, b \rangle = \alpha'^{\mathsf{gd}}(\eta_2)\sigma$ in

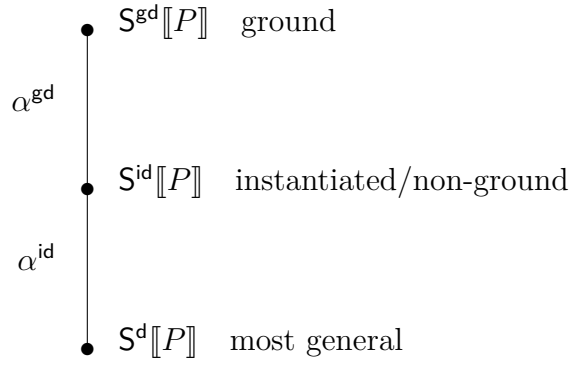
$$\alpha^{\mathsf{gd}}(\langle [\vdash p(v)],\, \varepsilon \rangle)\sigma \triangleq [\vdash \sigma(p(v))]$$

$$\begin{aligned}
\alpha^{\mathsf{gd}}(\varpi_1 \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}/\vartheta\rangle\!\rangle^{\mathsf{t}}} \eta_2 \xrightarrow{\ell} \theta)\sigma &\triangleq \varpi_1 \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}/\vartheta'\rangle\!\rangle^{\mathsf{t}}} \alpha^{\mathsf{gd}}(\varpi'_2 \xrightarrow{\ell} \theta)\sigma && \text{if } b \wedge \vartheta' \in \vartheta \uparrow \sigma \\
&\triangleq \varpi_1 && \text{if } \neg b \vee \vartheta' \notin \vartheta \uparrow \sigma \\
\alpha^{\mathsf{gd}}(\varpi_1 \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}\rangle\!\rangle^{\mathsf{t}}} \eta_2 \xrightarrow{\ell} \theta)\sigma &\triangleq \varpi_1 \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}\rangle\!\rangle^{\mathsf{t}}} \alpha^{\mathsf{gd}}(\varpi'_2 \xrightarrow{\ell} \theta)\sigma && \text{if } b \\
&\triangleq \varpi_1 && \text{if } \neg b \\
\alpha^{\mathsf{gd}}(\varpi \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}\rangle\!\rangle^{\mathsf{t}}} \langle [\dashv\!\square], \vartheta \rangle)\sigma &\triangleq \varpi \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}\rangle\!\rangle^{\mathsf{t}}} \langle [\dashv\!\square], \vartheta' \rangle && \text{if } \vartheta' \in \vartheta \uparrow \sigma \\
&\triangleq \varpi && \text{if } \vartheta' \notin \vartheta \uparrow \sigma
\end{aligned}$$

$$\alpha^{\mathsf{gd}}(\Theta) \triangleq \{\alpha^{\mathsf{gd}}(\theta)\sigma \mid \theta \in \Theta \wedge \sigma \in \overline{\mathbb{S}}\} \quad .$$

11.2.3 The Derivation Instantiation Abstraction Hierarchy

By instantiating most general maximal derivation semantics, we get a second dimension in our hierarchy of semantics relative to the degree of instantiation of the initial goal.



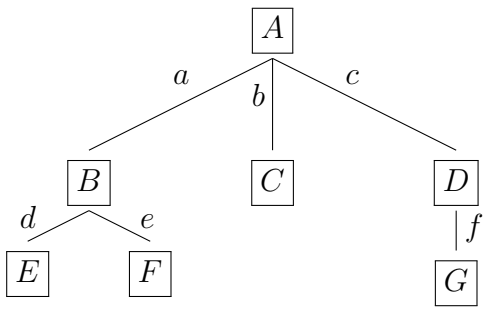
Of course, this can be combined with partial correctness abstractions. For example Herbrand models abstract away from finite failures and are relative to ground derivations only.

11.3 The Computational Information Abstractions

A third dimension abstracts away from the detailed information gathered by derivations on the computations. The abstraction below gets rid of information on computation, independently of partial correctness and instantiation abstractions, so it is a third dimension in the hierarchy of abstractions. Not all possible computational information abstractions have been considered here, our aim is to provide a small representative panel only.

11.3.1 The SLD-abstraction

The *SLD-abstraction* records the set of derivations for a goal in the form of a SLD-tree (as in [39,28] but keeping in addition the answer substitution). We encode trees in parenthesized form through a prefix traversal



$$\boxed{A}\,[\![\, a\,\boxed{B}\,[\![d\,\boxed{E}\,[\![]\!];e\,\boxed{F}\,[\![]\!]]\!];\quad b\,\boxed{C}\,[\![]\!];\quad c\,\boxed{D}\,[\![f\,\boxed{G}\,[\![]\!]]\!]$$

so that the syntax of SLD-trees $\xi \in \Xi$ is ($n \geqslant 1$)

$$\begin{array}{lll}
\xi ::= & \boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![\mathtt{i}_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\ \xi_1; \ldots ; \mathtt{i}_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\ \xi_n]\!] & \text{SLD derivation} \\
\mid & \boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![]\!] & \text{failure} \\
\mid & \boxed{\boxed{\sigma}}\ [\![]\!] & \text{success}
\end{array}$$

The contradiction $\boxed{\boxed{\sigma}}$ in the refutation contains the answer substitution σ. A forest is an indexed family $\langle \xi_i,\ i \in \Delta \rangle$ of SLD-trees ξ_i, $i \in \Delta$. They naturally arise in a PROLOG interpreter when considering a sequence of goals (instead of a set of goals).

The SLD-abstraction collects the nodes of the SLD-tree from the states of traces.

$$\begin{array}{rcl}
\alpha^{\mathsf{K}}(\langle [\vdash A],\ \vartheta \rangle) & \triangleq & \boxed{\leftarrow \vartheta(A)/\vartheta} \\
\alpha^{\mathsf{K}}(\langle \varpi,\ \vartheta \rangle) & \triangleq & \boxed{\leftarrow \langle \alpha'^{\mathsf{K}}(\langle \varpi,\ \vartheta \rangle),\ \vartheta \rangle}
\end{array}$$

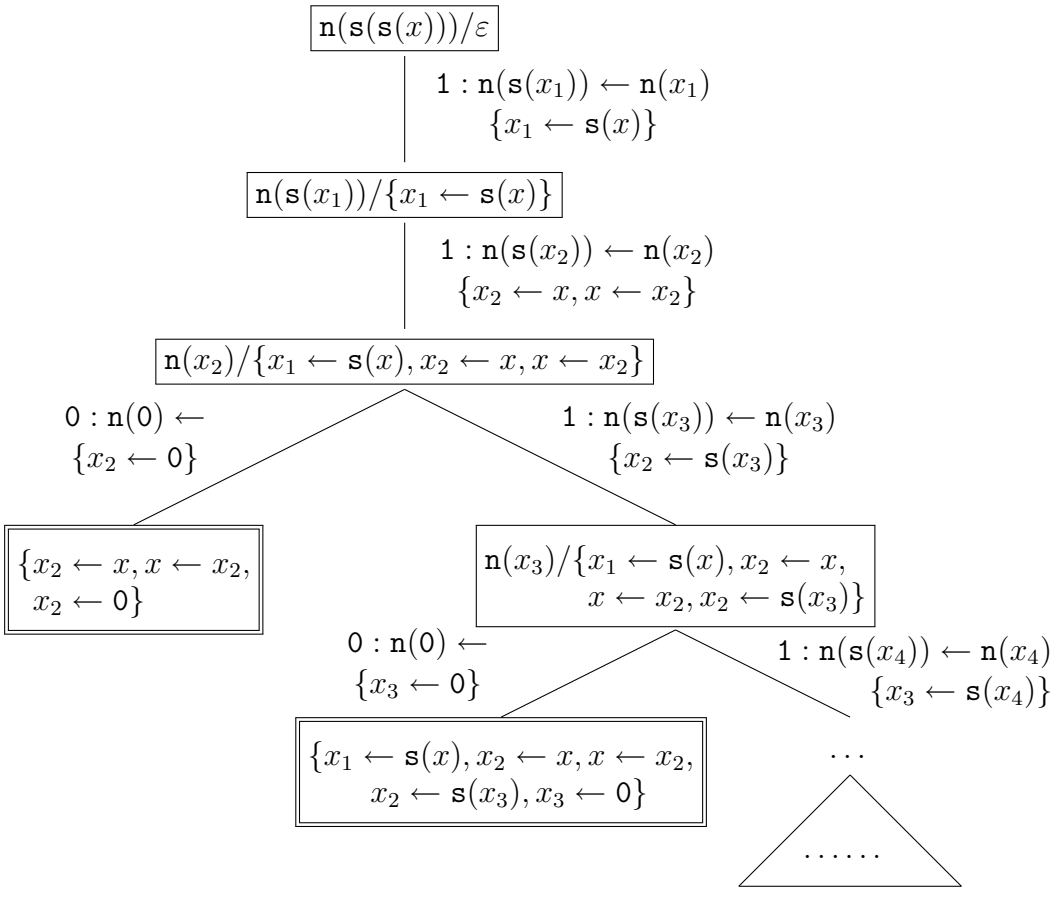
α'^{K} collects pending subgoals in inverse order on the stack.

$$\alpha'^{\mathsf{K}}(\langle\varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}.B\boldsymbol{B}'],\ \vartheta\rangle) \triangleq \vartheta(B\boldsymbol{B}')\alpha'^{\mathsf{K}}(\langle\varpi,\ \vartheta\rangle)$$
$$\alpha'^{\mathsf{K}}(\langle[\dashv\Box],\ \vartheta\rangle) \triangleq \boldsymbol{\varepsilon}$$

The SLD-trees are built from traces by grouping their common prefixes in the order of the PROLOG program clauses.

$$\alpha^{\mathsf{K}}(\Theta) \triangleq \{\alpha^{\mathsf{K}}(\eta)[\![\mathtt{i}_1:\ell_1\alpha^{\mathsf{K}}(\Theta_1);\ldots;\mathtt{i}_n:\ell_n\alpha^{\mathsf{K}}(\Theta_n)]\!] \mid \eta \in \mathscr{E} \wedge \mathtt{i}_1 < \ldots < \mathtt{i}_n \wedge$$
$$\Theta.\eta = \bigcup_{k=1}^{n} \Theta_k \wedge \forall k \in [1,n] : \Theta_k = \{\theta \mid \eta \xrightarrow{(\!|\mathtt{i}_k:\ell_k} {}^{\mathtt{t}}\ \theta \in \Theta.\eta\} \neq \varnothing\} \cup$$
$$\alpha^{\mathsf{K}}(\{\theta \mid \eta \xrightarrow{\mathtt{i}:C|\!)} {}^{\mathtt{t}}\ \theta \in \Theta\}) \cup \{\boxed{\boxed{\vartheta}}\ [\![]\!] \mid \exists\vartheta : \langle[\dashv\Box],\ \vartheta\rangle \in \Theta\} \quad .$$

Example 8 An SLD-derivation tree for the PROLOG program of **Ex. 1** is



□

α^{K} can be easily extended to ground derivations as was done in **Sect. 11.2** for traces.

11.3.2 The PROLOG *abstraction*

The PROLOG *abstraction* abstracts a forest $\langle \xi_i,\ i \in \Delta \rangle$ of SLD-trees ξ_i, $i \in \Delta$ into the set of execution traces corresponding to a depth-first traversal of these SLD-trees ξ_i (as in the PROLOG interpreter [40]). SLD-trees may have infinite branches so the execution sequence, defined by transfinite recursion, may be transfinite (and is truncated to ω by PROLOG interpreters, which is a further abstraction).

$$\alpha^{\mathsf{C}}(\langle \xi_i,\ i \in \Delta \rangle) \triangleq \langle \alpha^{\mathsf{C}}(\xi_i),\ i \in \Delta \rangle$$

$$\alpha^{\mathsf{C}}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!]) \triangleq$$
$$\boxed{\leftarrow \boldsymbol{B}/\sigma}\ i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\alpha^{\mathsf{C}}(\xi_1) \ldots i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\alpha^{\mathsf{C}}(\xi_n)$$
$$\alpha^{\mathsf{C}}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![]\!]) \triangleq \epsilon$$
$$\alpha^{\mathsf{C}}(\boxed{\boxed{\sigma}}\ [\![]\!]) \triangleq \sigma \quad .$$

11.3.3 The cut abstraction

Many PROLOG implementations have a `cut` to trigger backtracking. Programs can have cuts (denoted `!`) on the right-handside of clauses. We assume cuts are kept "as is" in clauses by the transitional and maximal derivation semantics and by the SLD tree abstraction.

The *cut abstraction* $\alpha^{!\mathsf{n}}$ abstracts a forest $\langle \xi_i,\ i \in \Delta \rangle$ of SLD-trees ξ_i, $i \in \Delta$ into a (transfinite) execution sequence corresponding to a depth-first traversal of these SLD-trees ξ_i with *cut* (as in the PROLOG interpreter [41]). If the program has no cut, $\alpha^{!\mathsf{n}}$ boils down to α^{C}.

$$\alpha^{!\mathsf{n}}(\langle \xi_i,\ i \in \Delta \rangle) \triangleq \langle \alpha^{!\mathsf{n}}(\xi_i),\ i \in \Delta \rangle \quad .$$

We use $\alpha^{!\mathsf{n}}$ for non-deterministic traversal of the SLD-trees with backtracking. In nondeterministic mode, the SLD-tree is traversed in depth-first order, top-down, left to right.

$$\alpha^{!\mathsf{n}}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![]\!]) \triangleq \epsilon$$

$$\alpha^{!\mathsf{n}}(\boxed{\boxed{\sigma}}\ [\![]\!]) \triangleq \sigma$$

$$\alpha^{!\mathsf{n}}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\,[\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$
$$\triangleq \boxed{\leftarrow \boldsymbol{B}/\sigma}\, i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\alpha^{!\mathsf{n}}(\xi_1) \ldots i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\alpha^{!\mathsf{n}}(\xi_n) \qquad \text{if } ! \notin \boldsymbol{B}$$

We go into deterministic traversal mode the first time a clause with a cut is encountered in nondeterministic traversal mode. We use $\alpha^{!\mathsf{d}}$ for deterministic traversal of the SLD-trees with backtracking cut after the first success.

$$\alpha^{!\mathsf{n}}(\boxed{\leftarrow \boldsymbol{B}\,!\,\boldsymbol{B}'/\sigma}\,[\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$
$$\triangleq \mathit{let}\ \langle \pi, - \rangle = \alpha^{!\mathsf{d}}(\mathit{top}, \boxed{\leftarrow \boldsymbol{B}\,!\,\boldsymbol{B}'/\sigma}\,[\![\, i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$
$$\mathit{in}\ \pi$$

The deterministic depth-first traversal of the SLD-tree with $\alpha^{!\mathsf{d}}$ goes top-down, left to right until the first success. The deterministic traversal abstraction $\alpha^{!\mathsf{d}}$ returns *failure* if resolution failed and *success* when it succeeded so as to keep track of failures until the first success.

The deterministic traversal abstraction $\alpha^{!\mathsf{d}}$ has a marker parameter $\ell = \mathit{top}$ or *below* to distinguish the level of the first encountered clause with a cut. The level marker $\ell = \mathit{top}$ is used in deterministic mode when the first cut is encountered. The level marker ℓ is then set to *below* when traversing the SLD-trees at lower levels.

$$\alpha^{!\mathsf{d}}(\ell, \boxed{\leftarrow \boldsymbol{B}/\sigma}\,[\![]\!]) \triangleq \langle \epsilon, \mathit{failure} \rangle$$

$$\alpha^{!\mathsf{d}}(\ell, \boxed{\boxed{\sigma}}\,[\![]\!]) \triangleq \langle \sigma, \mathit{success} \rangle$$

$$\alpha^{!\mathsf{d}}(\ell, \boxed{\leftarrow \boldsymbol{B}/\sigma}\,[\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$
$$\triangleq \mathit{let}\ \langle\langle \pi_i,\ f_i\rangle,\ i = 1, \ldots, n\rangle \triangleq \langle \alpha^{!\mathsf{d}}(\mathit{below}, \xi_i),\ i = 1, \ldots, n\rangle\ \mathit{in}$$
$$\mathit{if}\ \bigwedge_{i=1}^{n}(f_i = \mathit{failure})\ \mathit{then}$$
$$\langle \boxed{\leftarrow \boldsymbol{B}/\sigma}\, i_1 : A_1 \leftarrow \boldsymbol{B}'_1/\vartheta_1\pi_1 \ldots i_n : A_n \leftarrow \boldsymbol{B}'_n/\vartheta_n\pi_n,\ \mathit{failure}\rangle$$
$$\mathit{else}$$
$$\langle \boxed{\leftarrow \boldsymbol{B}/\sigma}\, i_1 : A_1 \leftarrow \boldsymbol{B}'_1/\vartheta_1\pi_1 \ldots i_k : A_k \leftarrow \boldsymbol{B}'_k/\vartheta_k\pi_k,\ \mathit{success}\rangle$$
$$\mathit{where}\ \bigwedge_{i=1}^{k-1}(f_i = \mathit{failure}) \wedge (f_k = \mathit{success})$$

We go on in deterministic mode at lower levels where cuts are ignored. We also go on in deterministic mode at top level before the last cut. Indeed all cuts but the last one in the righthand side of a clause are useless hence ignored.

if $(\ell = top \implies \texttt{!} \in \boldsymbol{B})$ then

$$\alpha^{\mathsf{!d}}(\ell, \boxed{\leftarrow \texttt{!}\boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$
$$\triangleq \alpha^{\mathsf{!d}}(\ell, \boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$

We go back in nondeterministic traversal mode after the last cut in the top level clause.

if $\texttt{!} \notin \boldsymbol{B}$ then

$$\alpha^{\mathsf{!d}}(top, \boxed{\leftarrow \texttt{!}\boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!])$$
$$\triangleq \langle \alpha^{\mathsf{!n}}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\,\xi_n]\!]), \text{–}\rangle$$

Therefore the SLD-tree is traversed in depth-first order, top-down, left to right in nondeterministic mode with backtracks until a clause containing a cut is encountered. The SLD-tree traversal goes on with that clause in deterministic mode without backtrack and goes back to nondeterministic mode only after the last cut of the first clause with a cut encountered in the SLD-tree nondeterministic traversal.

Example 9 The cut semantics of the following program

$$\begin{array}{ll}
\texttt{0:} & \texttt{p}(x,y) \leftarrow \texttt{q}(x)\ \texttt{!}\ \texttt{r}(y) \\
\texttt{1:} & \texttt{q(a)} \leftarrow \\
\texttt{2:} & \texttt{q(b)} \leftarrow \\
\texttt{3:} & \texttt{r(c)} \leftarrow \\
\texttt{4:} & \texttt{r(d)} \leftarrow
\end{array}$$

contains exactly the two following executions

- $\boxed{\leftarrow \texttt{p}(x,y)/\varepsilon}\ \texttt{0:p}(x',y') \leftarrow \texttt{q}(x')\ \texttt{!}\ \texttt{r}(y')/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y\}$ $\boxed{\leftarrow \texttt{q}(x')/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y\}}\ \texttt{1:q(a)} \leftarrow /\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \texttt{a}\}\ \boxed{\leftarrow \texttt{r}(y')/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \texttt{a}\}}\ \texttt{3:r(c)} \leftarrow /\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \texttt{a}\}\ \{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \texttt{a}, y' \leftarrow \texttt{c}\}$
- $\boxed{\leftarrow \texttt{p}(x,y)/\varepsilon}\ \texttt{0:p}(x',y') \leftarrow \texttt{q}(x')\ \texttt{!}\ \texttt{r}(y')/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y\}$ $\boxed{\leftarrow \texttt{q}(x')/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y\}}\ \texttt{1:q(a)} \leftarrow /\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \texttt{a}\}\ \boxed{\leftarrow \texttt{r}(y')/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \texttt{a}\}}\ \texttt{4:r(d)} \leftarrow$

$/\{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \mathtt{a}\}\ \{x \leftarrow x', x' \leftarrow x, y \leftarrow y', y' \leftarrow y, x' \leftarrow \mathtt{a}, y' \leftarrow \mathtt{d}\}$ □

11.3.4 Lazy backtracking

Some implementations of PROLOG like the *Ciao Prolog System* [42] have lazy backtracking meaning that the system will backtrack only as necessary to obtain one solution (at the top level) and will not look for more solutions. This *lazy backtracking abstraction* α^{ℓ} abstracts a forest $\langle \xi_i,\ i \in \Delta \rangle$ of SLD-trees ξ_i, $i \in \Delta$ into a (transfinite) execution sequence corresponding to a depth-first traversal of these SLD-trees ξ_i until the first success at the top-level

$$\alpha^{\ell}(\langle \xi_i,\ i \in \Delta \rangle) \triangleq \langle \mathit{let}\ \langle \pi,\ - \rangle = \alpha^{\mathsf{Id}}(\mathit{top}, \xi_i)\ \mathit{in}\ \pi,\ i \in \Delta \rangle\ .$$

Example 10 The lazy backtracking semantics of the program of **Ex. 9** contains only the first of the two executions. □

11.3.5 The BF-semantics

The *breadth-first abstraction* explores the forest $\langle \xi_i,\ i \in \Delta \rangle$ by traversal of each tree $\xi_i, i \in \Delta$ in the forest level by level.

$$\alpha^{\mathsf{B}}(\langle \xi_i,\ i \in \Delta \rangle) \triangleq \langle \alpha^{\mathsf{B}_r}(\xi_i),\ i \in \Delta \rangle \alpha^{\mathsf{B}}(\langle \alpha^{\mathsf{B}_s}(\xi_i),\ i \in \Delta \rangle)$$

(where concatenation of sequences is denoted by juxtaposition). The exploration of the roots

$$\begin{aligned}
\alpha^{\mathsf{B}_r}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_n\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\ \xi_n]\!]) &\triangleq \\
\boxed{\leftarrow \boldsymbol{B}/\sigma}A_1 \leftarrow \boldsymbol{B}_1/\vartheta_n \ldots; A_n \leftarrow \boldsymbol{B}_n/\vartheta_n & \\
\alpha^{\mathsf{B}_r}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![]\!]) \triangleq \epsilon & \\
\alpha^{\mathsf{B}_r}(\boxed{\boxed{\sigma}}\ [\![]\!]) \triangleq \sigma &
\end{aligned}$$

is followed by the breadth-first exploration of the sons of each tree ξ_i, $i \in \Delta$

$$\alpha^{\mathsf{B}_s}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![i_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_n\xi_1; \ldots; i_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\ \xi_n]\!]) \triangleq \xi_1 \ldots \xi_n$$

11.3.6 The call-patterns abstraction

The *call-patterns abstraction* collects the goal, call-patterns and the answer substitution for each derivation, including those leading to finite failures [43].

$$\alpha^{\mathsf{p}}(\langle \xi_i,\ i \in \Delta \rangle) \triangleq \bigcup\{\alpha^{\mathsf{p}}(\xi_i) \mid i \in \Delta\} \qquad \text{SLD derivation forest}$$

$$\alpha^{\mathsf{p}}(\boxed{\leftarrow A/\sigma}\ [\![\mathtt{i}_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\ \xi_1; \ldots; \mathtt{i}_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\ \xi_n]\!]) \triangleq \qquad \text{SLD tree}$$
$$\alpha'^{\mathsf{p}}(\boxed{\leftarrow A/\sigma}\ [\![\mathtt{i}_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\ \xi_1; \ldots; \mathtt{i}_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\ \xi_n]\!])(\sigma(A))$$

$$\alpha'^{\mathsf{p}}(\boxed{\leftarrow B\boldsymbol{B}/\sigma}\ [\![\mathtt{i}_1 : A_1 \leftarrow \boldsymbol{B}_1/\vartheta_1\ \xi_1; \ldots; \mathtt{i}_n : A_n \leftarrow \boldsymbol{B}_n/\vartheta_n\ \xi_n]\!])A' \triangleq$$
$$\{\langle \sigma(A'),\ \sigma(B)\rangle\} \cup \bigcup_{i=1}^{n} \alpha'^{\mathsf{p}}(\xi_i)(A')$$

$$\alpha'^{\mathsf{p}}(\boxed{\leftarrow \boldsymbol{B}/\sigma}\ [\![]\!])A' \triangleq \varnothing \qquad \text{failure}$$

$$\alpha'^{\mathsf{p}}(\boxed{\boxed{\sigma}}\ [\![]\!])A' \triangleq \{\langle \sigma(A'),\ [\dashv\Box]\rangle\} \qquad \text{success.}$$

Combining with the α^{K} abstraction, this can also be understood as the following abstraction of the derivation semantics

$$\alpha^{\mathsf{p}}(\Theta) \triangleq \bigcup\{\alpha^{\mathsf{p}}(\theta) \mid \theta \in \Theta\}$$

$$\alpha^{\mathsf{p}}(\langle [\vdash A],\ \vartheta\rangle \xrightarrow{(\!|\,\mathtt{i}:A' \leftarrow \boldsymbol{B}/\vartheta}{}^{\mathsf{t}}\ \theta) \triangleq \{\langle \vartheta(A),\ \vartheta(A)\rangle\} \cup \alpha'^{\mathsf{p}}(\theta)\vartheta(A)$$

$$\alpha'^{\mathsf{p}}(\langle \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}.B\boldsymbol{B}'],\ \vartheta\rangle \xrightarrow{(\!|\,\mathtt{j}:B' \leftarrow \boldsymbol{B}''/\sigma}{}^{\mathsf{t}}\ \theta)A' \triangleq \{\langle \vartheta(A'),\ \vartheta(B)\rangle\} \cup \alpha'^{\mathsf{p}}(\theta)A'$$

$$\alpha'^{\mathsf{p}}(\langle \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}.],\ \vartheta\rangle \xrightarrow{\mathtt{i}:A \leftarrow \boldsymbol{B}\,|\!)}{}^{\mathsf{t}}\ \theta)A' \triangleq \alpha'^{\mathsf{p}}(\theta)A'$$

$$\alpha'^{\mathsf{p}}(\langle \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}.B\boldsymbol{B}'],\ \vartheta\rangle)A' \triangleq \varnothing \qquad \text{failure}$$

$$\alpha'^{\mathsf{p}}(\langle [\dashv\Box],\ \vartheta\rangle)A' \triangleq \{\langle \vartheta(A'),\ [\dashv\Box]\rangle\} \qquad \text{success.}$$

The above abstraction defines *success/correct call patterns* since finite failure are disregarded. An alternative is to consider *failure call patterns* by redefining

$$\alpha'^{\mathsf{p}}(\langle \varpi[\mathtt{i}:A \leftarrow \boldsymbol{B}.B\boldsymbol{B}'],\ \vartheta\rangle)A' \triangleq \langle \vartheta(A),\ [\dashv!]\rangle \qquad \text{failure}$$

where $[\dashv!]$ marks failure.

Example 11 For the following PROLOG programs, we have

$0{:}\mathtt{p}(\mathtt{a}) \leftarrow$

$1{:}\mathtt{p}(x) \leftarrow$

$2{:}\mathtt{q}(x) \leftarrow \mathtt{p}(x)$

$$\mathsf{S}^{\mathsf{p}}[\![P]\!] \triangleq \{\langle \mathtt{p}(\mathtt{a}),\, \mathtt{p}(\mathtt{a})\rangle, \langle \mathtt{p}(\mathtt{a}),\, [\dashv\Box]\rangle \langle \mathtt{p}(x),\, \mathtt{p}(x)\rangle, \langle \mathtt{p}(x),\, [\dashv\Box]\rangle \langle \mathtt{q}(\mathtt{a}),\, \mathtt{q}(\mathtt{a})\rangle, \langle \mathtt{q}(\mathtt{a}),\, \mathtt{p}(\mathtt{a})\rangle, \langle \mathtt{q}(\mathtt{a}),\, [\dashv\Box]\rangle, \langle \mathtt{q}(x),\, \mathtt{q}(x)\rangle, \langle \mathtt{q}(x),\, \mathtt{p}(x)\rangle, \langle \mathtt{q}(x),\, [\dashv\Box]\rangle\}$$

$0{:}\mathtt{p}(x) \leftarrow$

$1{:}\mathtt{q}(x) \leftarrow \mathtt{p}(x)$

$$\mathsf{S}^{\mathsf{p}}[\![P']\!] \triangleq \{\langle \mathtt{p}(x),\, \mathtt{p}(x)\rangle, \langle \mathtt{p}(x),\, [\dashv\Box]\rangle \langle \mathtt{q}(x),\, \mathtt{q}(x)\rangle, \langle \mathtt{q}(x),\, \mathtt{p}(x)\rangle \langle \mathtt{q}(x),\, [\dashv\Box]\rangle\}$$

□

11.3.7 The Model Abstraction

The *model abstraction* collects answers in the call patterns

$$\alpha^{\mathsf{m}}(K) \triangleq \{A \in \mathbb{A} \mid \langle A,\, [\dashv\Box]\rangle \in K\}$$

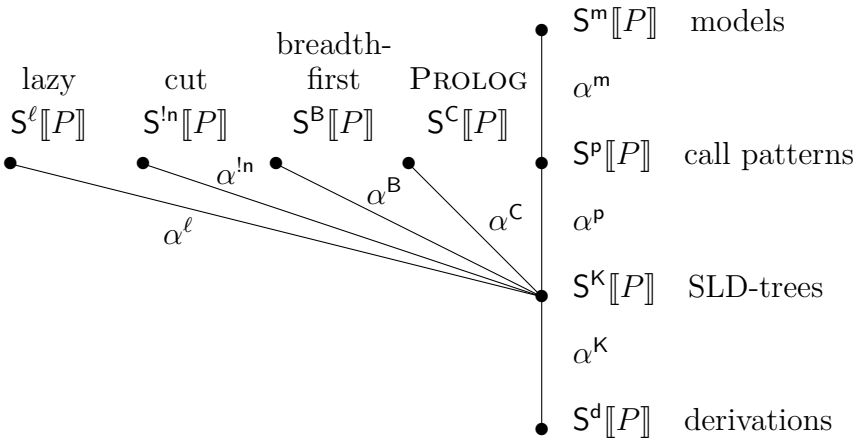
Example 12 For **Ex. 11**, we have

$$\mathsf{S}^{\mathsf{m}}[\![P]\!] \triangleq \{\mathtt{p}(\mathtt{a}), \mathtt{p}(x), \mathtt{q}(\mathtt{a}), \mathtt{q}(x)\} \qquad \mathsf{S}^{\mathsf{m}}[\![P']\!] \triangleq \{\mathtt{p}(x), \mathtt{q}(x)\}$$

□

11.3.8 The Computational Information Abstraction Hierarchy

The third dimension in the hierarchy is the following



(where the composition of partial correctness abstractions with α^{C} leads to non-computable semantics but are useful when reasoning on program implementations).

11.4 The Hierarchy of Abstractions and Semantics

The combination of the instantiation abstraction of **Sec. 11.2.3** and the information abstraction of **Sec. 11.3.8** yields to the two-dimensional hierarchy of abstractions of **Fig. 1**. Missing in the picture is the partial correctness third abstraction dimension of **Sec. 11.1.2**.

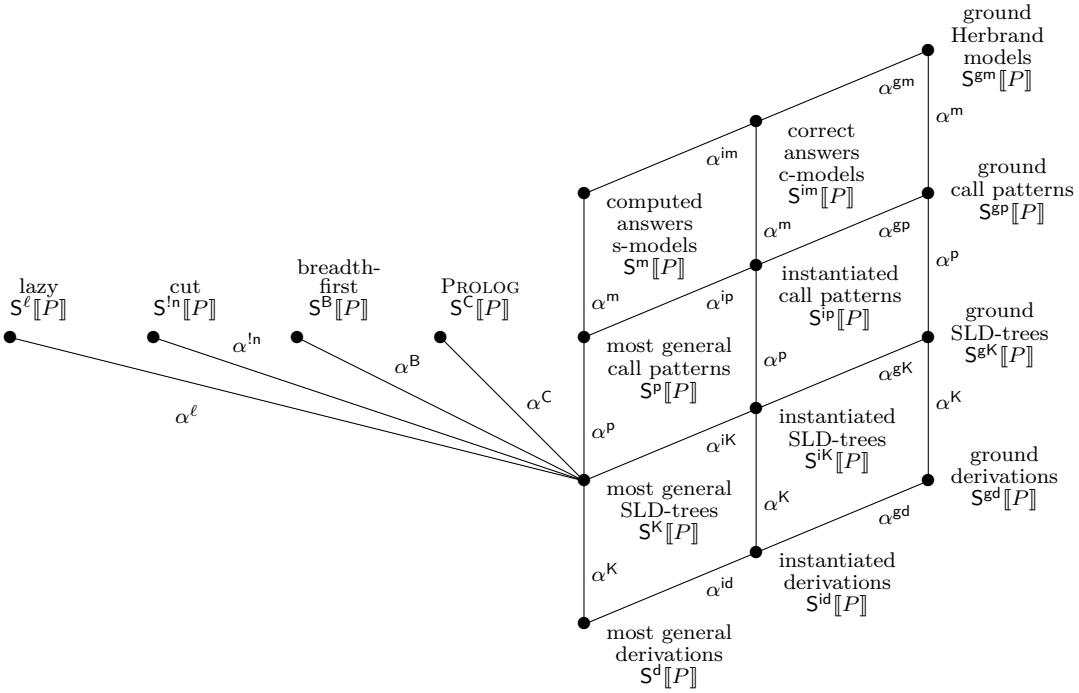


Fig. 1. The hierarchy of maximal abstractions

By applying this hierarchy of abstractions to the most general maximal derivation semantics $\mathsf{S}^{\mathsf{d}}[\![P]\!]$, we get the hierarchy of maximal semantics given in **Fig. 2**. Classical examples in the hierarchy of semantics is given in **Fig. 3**, some of which are detailed below.

11.4.1 The s-semantics

The *s*-semantics $\mathsf{S}^{\mathsf{s}}[\![P]\!]$ provides *computed answers* [46]:

$$\mathsf{S}^{\mathsf{s}}[\![P]\!] \triangleq \alpha^{\mathsf{ds}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])$$

	derivations	SLD-trees	call patterns	models
ground	$\mathsf{S}^{\mathsf{gd}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{gK}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{gp}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{gm}}[\![P]\!] \triangleq$
	$\alpha^{\mathsf{gd}}(\mathsf{S}^{\mathsf{id}}[\![P]\!])$	$\alpha^{\mathsf{gK}}(\mathsf{S}^{\mathsf{gd}}[\![P]\!])$	$\alpha^{\mathsf{gp}}(\mathsf{S}^{\mathsf{gK}}[\![P]\!])$	$\alpha^{\mathsf{gm}}(\mathsf{S}^{\mathsf{gp}}[\![P]\!])$
instantiation	$\mathsf{S}^{\mathsf{id}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{iK}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{ip}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{im}}[\![P]\!] \triangleq$
	$\alpha^{\mathsf{id}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])$	$\alpha^{\mathsf{iK}}(\mathsf{S}^{\mathsf{id}}[\![P]\!])$	$\alpha^{\mathsf{ip}}(\mathsf{S}^{\mathsf{iK}}[\![P]\!])$	$\alpha^{\mathsf{im}}(\mathsf{S}^{\mathsf{ip}}[\![P]\!])$
most general	$\mathsf{S}^{\mathsf{d}}[\![P]\!]$	$\mathsf{S}^{\mathsf{K}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{p}}[\![P]\!] \triangleq$	$\mathsf{S}^{\mathsf{m}}[\![P]\!] \triangleq$
		$\alpha^{\mathsf{K}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])$	$\alpha^{\mathsf{p}}(\mathsf{S}^{\mathsf{K}}[\![P]\!])$	$\alpha^{\mathsf{m}}(\mathsf{S}^{\mathsf{p}}[\![P]\!])$

Fig. 2. The hierarchy of maximal semantics

$\mathsf{S}^{\mathsf{H}}[\![P]\!] \triangleq \alpha^{\mathsf{m}}(\alpha^{\mathsf{p}}(\alpha^{\mathsf{K}}(\alpha^{\mathsf{sd}}(\alpha^{\mathsf{gd}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])))))$	minimal Herbrand-model semantics (logical consequences) of Maarten van Emden and Robert Kowalski [1]
$\mathsf{S}^{\mathsf{c}}[\![P]\!] \triangleq \alpha^{\mathsf{m}}(\alpha^{\mathsf{p}}(\alpha^{\mathsf{K}}(\alpha^{\mathsf{sd}}(\alpha^{\mathsf{id}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])))))$	c-semantics (correct answer substitutions) of Keith Clark [44,45]
$\mathsf{S}^{\mathsf{s}}[\![P]\!] \triangleq \alpha^{\mathsf{m}}(\alpha^{\mathsf{p}}(\alpha^{\mathsf{K}}(\alpha^{\mathsf{sd}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])))))$	s-semantics (computed answer substitutions) of Giorgio Levi [46]
$\mathsf{S}^{\mathsf{sp}}[\![P]\!] \triangleq \alpha^{\mathsf{p}}(\alpha^{\mathsf{K}}(\alpha^{\mathsf{sd}}(\mathsf{S}^{\mathsf{d}}[\![P]\!]))))$	correct call patterns of Maurizio Gabbrielli and Roberto Giacobazzi [43]
$\mathsf{S}^{\mathsf{p}}[\![P]\!] \triangleq \alpha^{\mathsf{p}}(\alpha^{\mathsf{K}}(\mathsf{S}^{\mathsf{d}}[\![P]\!]))$	call patterns of Maurizio Gabbrielli, Giorgio Levi and Maria Chiara Meo [47]
$\mathsf{S}^{\mathsf{iK}}[\![P]\!] \triangleq \alpha^{\mathsf{K}}(\alpha^{\mathsf{id}}(\mathsf{S}^{\mathsf{d}}[\![P]\!]))$	maximal SLD-trees of Robert Kowalski [39,28]

Fig. 3. Examples of classical semantics in the hierarchy

where $\alpha^{\mathsf{ds}} \triangleq \alpha^{\mathsf{m}} \circ \alpha^{\mathsf{p}} \circ \alpha^{\mathsf{K}} \circ \alpha^{\mathsf{sd}}$ is

$$\alpha^{\mathsf{ds}}(\langle[\vdash \mathtt{p}(v)],\, \varepsilon\rangle \xrightarrow{\ell}{}^{\mathsf{t}}\, \theta \xrightarrow{\ell'}{}^{\mathsf{t}}\, \langle[\dashv\Box],\, \vartheta\rangle) = \{\vartheta(\mathtt{p}(v))\} \qquad \theta \in \mathbf{\Theta}^*$$

$$\alpha^{\mathsf{ds}}(\Theta) = \bigcup\{\alpha^{\mathsf{ds}}(\theta) \mid \theta \in \Theta\}$$

The ordering of the program clauses is lost as well as the finite failures and infinite behaviors.

Example 13 For both PROLOG programs of **Ex. 11**, we have

$$\mathsf{S}^{\mathsf{s}}[\![P]\!] \triangleq \{\mathtt{p}(\mathtt{a}), \mathtt{p}(x), \mathtt{q}(\mathtt{a}), \mathtt{q}(x)\} \qquad \mathsf{S}^{\mathsf{s}}[\![P']\!] \triangleq \{\mathtt{p}(x), \mathtt{q}(x)\}$$

□

11.4.2 The c-semantics

The *c-semantics* $\mathsf{S}^{\mathsf{c}}[\![P]\!]$ provides *correct answers* [44,45]

$$\mathsf{S}^{\mathsf{c}}[\![P]\!] \triangleq \alpha^{\mathsf{dc}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])$$

where $\alpha^{\mathsf{dc}} \triangleq \alpha^{\mathsf{m}} \circ \alpha^{\mathsf{p}} \circ \alpha^{\mathsf{K}} \circ \alpha^{\mathsf{sd}} \circ \alpha^{\mathsf{id}}$ is

$$\alpha^{\mathsf{dc}}(\langle[\vdash \mathtt{p}(v)],\, \varepsilon\rangle \xrightarrow{\ell}{}^{\mathsf{t}}\, \theta \xrightarrow{\ell'}{}^{\mathsf{t}}\, \langle[\dashv\Box],\, \vartheta\rangle) = \{\sigma(\vartheta(\mathtt{p}(v))) \mid \sigma \in \mathbb{S}\} \qquad \theta \in \mathbf{\Theta}^*$$
$$\alpha^{\mathsf{dc}}(\Theta) = \bigcup\{\alpha^{\mathsf{dc}}(\theta) \mid \theta \in \Theta\}$$

Example 14 For both PROLOG programs of **Ex. 11**, we have

$$\mathsf{S}^{\mathsf{c}}[\![P]\!] \triangleq \mathsf{S}^{\mathsf{c}}[\![P']\!] \triangleq \{\mathtt{p}(\mathtt{a}), \mathtt{p}(x), \mathtt{q}(\mathtt{a}), \mathtt{q}(x)\}$$

□

11.4.3 The H-semantics

The H-semantics $\mathsf{S}^{\mathsf{H}}[\![P]\!]$ provides the *least Herbrand model* of the PROLOG program P [1]

$$\mathsf{S}^{\mathsf{H}}[\![P]\!] \triangleq \alpha^{\mathsf{dH}}(\mathsf{S}^{\mathsf{d}}[\![P]\!])$$

where $\alpha^{\mathsf{dH}} \triangleq \alpha^{\mathsf{m}} \circ \alpha^{\mathsf{p}} \circ \alpha^{\mathsf{K}} \circ \alpha^{\mathsf{sd}} \circ \alpha^{\mathsf{gd}}$ is

$$\alpha^{\mathsf{dH}}(\langle[\vdash \mathtt{p}(v)],\, \varepsilon\rangle \xrightarrow{\ell}{}^{\mathsf{t}}\, \theta \xrightarrow{\ell'}{}^{\mathsf{t}}\, \langle[\dashv\Box],\, \vartheta\rangle) = \{\sigma(\vartheta(\mathtt{p}(v))) \mid \sigma \in \overline{\mathbb{S}}\} \qquad \theta \in \mathbf{\Theta}^*$$
$$\alpha^{\mathsf{dH}}(\Theta) = \bigcup\{\alpha^{\mathsf{dH}}(\theta) \mid \theta \in \Theta\}$$

Example 15 For both PROLOG programs of **Ex. 13**, we have

$$\mathsf{S}^{\mathsf{H}}[\![P]\!] \triangleq \{\mathtt{p}(\mathtt{a}), \mathtt{q}(\mathtt{a})\}$$

□

12 Fixpoint Bottom-Up Semantics

We now show that the bottom-up most general maximal derivation semantics can be expressed un fixpoint form. Because this property is preserved by abstraction, all semantics in the hierarchy of bottom-up semantics can also be expressed in fixpoint form, a property which, by further abstractions, can be exploited in static program analysis.

12.1 Inlaying

For the recursive *inlay* of a derivation into another one, we need the operation

$$\begin{array}{l}\langle \varpi,\, \varpi',\, \varsigma\rangle \Uparrow^{\mathsf{d}} \langle [\vdash A],\, \boldsymbol{\sigma}_0\rangle \xrightarrow{\ell_0} \langle [\dashv\Box]\varpi_1,\, \boldsymbol{\sigma}_1\rangle \ldots \langle [\dashv\Box]\varpi_n,\, \boldsymbol{\sigma}_n\rangle \xrightarrow{\ell_n} \langle [\dashv\Box],\, \boldsymbol{\sigma}_{n+1}\rangle \\ \quad \triangleq\ \varsigma\big(\langle \varpi,\, \boldsymbol{\sigma}_0\rangle \xrightarrow{\ell_0} \langle \varpi'\varpi_1,\, \boldsymbol{\sigma}_1\rangle \ldots \langle \varpi'\varpi_n,\, \boldsymbol{\sigma}_n\rangle \xrightarrow{\ell_n} \langle \varpi',\, \boldsymbol{\sigma}_{n+1}\rangle\big)\end{array}$$

where the application of the substitution expression to the trace is defined as

$$\begin{array}{l}\varsigma\big(\langle \varpi_0,\, \boldsymbol{\sigma}_0\rangle \xrightarrow{\ell_0} \langle \varpi_1,\, \boldsymbol{\sigma}_1\rangle \ldots \langle \varpi_n,\, \boldsymbol{\sigma}_n\rangle \xrightarrow{\ell_n} \langle \varpi_{n+1},\, \boldsymbol{\sigma}_{n+1}\rangle\big) \\ \quad \triangleq\ \vec{\epsilon} \\ \qquad \text{if } \boldsymbol{\sigma}_0 \upharpoonright \varsigma = \emptyset \\ \quad \triangleq\ \langle \varpi_0,\, \varsigma'_0\rangle \xrightarrow{\ell_0} \langle \varpi_1,\, \varsigma'_1\rangle \ldots \langle \varpi_{k-1},\, \varsigma'_{k-1}\rangle \xrightarrow{\ell_{k-1}} \langle \varpi_k,\, \varsigma_k\rangle \\ \qquad \text{if } \varsigma'_0 = \boldsymbol{\sigma}_0 \upharpoonright \varsigma \neq \emptyset \wedge \ldots \wedge \varsigma'_k = \boldsymbol{\sigma}_k \upharpoonright \varsigma \neq \emptyset \wedge \boldsymbol{\sigma}'_{k+1} \upharpoonright \varsigma = \emptyset \\ \quad \triangleq\ \langle \varpi_0,\, \varsigma'_0\rangle \xrightarrow{\ell_0} \langle \varpi_1,\, \varsigma'_1\rangle \ldots \langle \varpi_n,\, \varsigma'_n\rangle \xrightarrow{\ell_n} \langle \varpi_{n+1},\, \varsigma'_{n+1}\rangle \\ \qquad \text{if } \varsigma'_0 = \boldsymbol{\sigma}_0 \upharpoonright \varsigma \neq \emptyset \wedge \ldots \wedge \varsigma'_{n+1} = \boldsymbol{\sigma}_{n+1} \upharpoonright \varsigma \neq \emptyset\end{array}$$

with standardization apart, and

$$\langle \varpi,\, \varpi',\, \varsigma\rangle \Uparrow^{\mathsf{d}} \Theta \triangleq \{\langle \varpi,\, \varpi',\, \varsigma\rangle \Uparrow^{\mathsf{d}} \theta \mid \theta \Subset \Theta\}\ . \tag{8}$$

Example 16 A trace for the goal $\mathtt{n}(x)$ with $x = \mathtt{0}$ as defined by the logic program of **Ex. 1** is:

$$\begin{array}{l}\theta_0 \triangleq \langle [\vdash \mathtt{n}(x')],\, \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow/\{x'\leftarrow \mathtt{0}\}}{}^{\mathsf{pt}} \langle [\dashv\Box][\mathtt{0}:\mathtt{n}(x') \leftarrow \centerdot],\, \{x' \leftarrow \mathtt{0}\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow |\!\rangle}{}^{\mathsf{pt}} \\ \qquad \langle [\dashv\Box],\, \{x' \leftarrow \mathtt{0}\}\rangle\end{array}$$

A trace θ_1 for the goal $\mathtt{n}(x)$ with $x = \mathtt{s}(\mathtt{0})$ as defined by the PROLOG program of **Ex. 1** is obtained by inlay of θ_0:

$$\theta_1 \triangleq \langle[\vdash \mathtt{n}(x)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')/\{x\leftarrow\mathtt{s}(x')\}}{}^{\mathsf{pt}} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\ \leftarrow\ \boldsymbol{.}\mathtt{n}(x')],$$
$$[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\ \leftarrow\ \mathtt{n}(x')\boldsymbol{.}],\ \{x\leftarrow\mathtt{s}(x')\}\rangle \Uparrow^{\mathsf{d}} \theta_0 \xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\,|\!\rangle}{}^{\mathsf{pt}} \langle[\dashv\square],$$
$$\{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{0}\}\rangle$$

where

$$\langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\boldsymbol{.}\mathtt{n}(x')],\ [\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\boldsymbol{.}],\ \{x\leftarrow\mathtt{s}(x')\}\rangle \Uparrow^{\mathsf{d}} \theta_0$$

$$= \quad \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\boldsymbol{.}\mathtt{n}(x')],\ \{x\leftarrow\mathtt{s}(x')\}\rangle$$
$$\xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow/\{x'\leftarrow\mathtt{0}\}}{}^{\mathsf{pt}}$$
$$\langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\boldsymbol{.}][\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow\boldsymbol{.}],\ \{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{0}\}\rangle$$
$$\xrightarrow{\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow|\!\rangle}{}^{\mathsf{pt}}$$
$$\langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x'))\leftarrow\mathtt{n}(x')\boldsymbol{.}],\ \{x\leftarrow\mathtt{s}(x'),x'\leftarrow\mathtt{0}\}\rangle$$

□

12.2 Fixpoint Bottom-Up Most General Maximal Derivation Semantics

Let us define the *bottom-up set of traces transformer* $\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!] \in \wp(\boldsymbol{\Theta}) \mapsto \wp(\boldsymbol{\Theta})$ for a PROLOG program $P \in \mathbb{P}$ as

$$\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!] \triangleq \boldsymbol{\lambda}\,\Theta\bullet \bigcup_{\mathtt{i}:A\leftarrow\boldsymbol{B}\notin P,\, p\in\mathbb{p},\, v\in\mathbb{V},\, \vartheta\in mgu(p(v),A)} \langle[\vdash p(v)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow\boldsymbol{B}/\vartheta} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow\boldsymbol{.B}]\,\vartheta\,\Theta \qquad (9)$$

where the *clause transformer* $\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\ \leftarrow\ \boldsymbol{B.B'}] \in \mathbb{S} \mapsto \wp(\boldsymbol{\Theta}) \mapsto \wp(\boldsymbol{\Theta})$ is defined as

$$\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow\boldsymbol{B.}B\boldsymbol{B'}] \triangleq \boldsymbol{\lambda}\,\vartheta\bullet\boldsymbol{\lambda}\,\Theta\bullet \qquad (10)$$
$$\{(\langle[\dashv\square][\mathtt{i}:A\leftarrow\boldsymbol{B.}B\boldsymbol{B'}],\ [\dashv\square][\mathtt{i}:A\leftarrow\boldsymbol{B}B\boldsymbol{.B'}],\ \vartheta\rangle \Uparrow^{\mathsf{d}} \eta \xrightarrow{\ell} \langle\varpi,\ \vartheta'\rangle)\fatsemi\theta \mid$$
$$\eta \xrightarrow{\ell} \langle\varpi,\vartheta'\rangle \in \Theta.B' \wedge \sigma \in mgu(B,B') \wedge \theta \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow\boldsymbol{B}B\boldsymbol{.B'}]\,(\vartheta\uparrow\sigma\uparrow\vartheta'^{\,3})\,\Theta\}$$

$$\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow\boldsymbol{B.}] \triangleq \boldsymbol{\lambda}\,\vartheta\bullet\boldsymbol{\lambda}\,\Theta\bullet\{\langle[\dashv\square][\mathtt{i}:A\leftarrow\boldsymbol{B.}],\ \vartheta\rangle \xrightarrow{\mathtt{i}:A\leftarrow\boldsymbol{B}\,|\!\rangle} \langle[\dashv\square],\ \vartheta\rangle\}\ . \qquad (11)$$

Lemma 17 *For all programs P, $\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!]$ and for all definite clause states $[\boldsymbol{i}:A\ \leftarrow\ \boldsymbol{B.B'}]$ and subsitutions ϑ, $\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\boldsymbol{i}:A\ \leftarrow\ \boldsymbol{B.B'}]\vartheta$, are complete join morphisms.* □

PROOF Additivity directly follows from (9) for $\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!]$. For $\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow\boldsymbol{B.B'}]\,\vartheta$, this is obvious in case (11) and follows by induction for case (10). ■

[3] Note that the composition $\uparrow$ of substitutions is associative and commutative.

Lemma 18 *If all traces in $T \subseteq \mathbf{\Theta}$ are derivations of the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ then all traces in $\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}{:}A \leftarrow \boldsymbol{B}.\boldsymbol{B}']\vartheta T$ are generated by the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$, start in state $\langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}.\boldsymbol{B}'],\ \vartheta\rangle$ and end in a final state in $\mathscr{E}^{AS} \cup \mathscr{E}^{FF}$.* □

PROOF The proof is by induction on the length of $\boldsymbol{B}'$.

Base case: if $\boldsymbol{B}' = \boldsymbol{\varepsilon}$ then the trace is $\langle\varpi[\mathtt{i}{:}A \leftarrow \boldsymbol{B}.],\ \vartheta\rangle \xrightarrow{\mathtt{i}:A\leftarrow\boldsymbol{B}\rangle\!\rangle^{\mathtt{t}}} \langle\varpi,\ \vartheta\rangle$ where $\mathtt{i}{:}A \leftarrow \boldsymbol{B} \in P$, which, by (4), is a correct trace generated by $\mathsf{S}^{\mathsf{t}}[\![P]\!]$.

Inductive case: if $\boldsymbol{B}' = B\boldsymbol{B}''$ then (10) applies. By hypothesis, all traces in $T \subseteq \mathbf{\Theta}$ are derivations of the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ hence so are those in the subset $T.B$. All these traces have the form

$$\begin{aligned}\theta = \langle[\vdash B],\ \varepsilon\rangle &\xrightarrow{\langle\!|\mathtt{k}:B'\leftarrow\boldsymbol{B}/\sigma} \langle[\dashv\square][\mathtt{k}{:}B \leftarrow \boldsymbol{B}.],\ \vartheta_1\rangle \xrightarrow{\ell_1} \\ &\langle[\dashv\square]\varpi_2,\ \vartheta_2\rangle \xrightarrow{\ell_2} \ldots \xrightarrow{\ell_{n-2}} \langle[\dashv\square]\varpi_{n-1},\ \vartheta_{n-1}\rangle \xrightarrow{\ell_{n-1}} \eta_n\end{aligned}$$

where

$$\begin{aligned}\eta_n &= \langle[\dashv\square],\ \vartheta_n\rangle && \text{if the computation succeeds} \\ &= \langle[\dashv\square]\varpi_n,\ \vartheta_n\rangle \not\longrightarrow && \text{for finite failure.}\end{aligned}$$

Because trace inlaying preserves finite failure, the only possible cases which can continue the computation are inlaying of successfull traces. The case of traces ending in $\mathscr{E}^{\mathrm{FF}}$ is therefore straightforward. Consequently, we only consider the traces in the subset $T.B$ ending in a state in $\mathscr{E}^{\mathrm{AS}}$.

In this case, we have

$$\langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}.\overline{B}\boldsymbol{B}'],\ [\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}\overline{B}.\boldsymbol{B}'],\ \vartheta\rangle \Uparrow^{\mathsf{d}} \theta = \theta'$$

where
$\theta' = \langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}.\overline{B}\boldsymbol{B}'],\ \vartheta'\rangle \xrightarrow{\langle\!|\mathtt{k}:B'\leftarrow\boldsymbol{B}/\sigma} \langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}\overline{B}.\boldsymbol{B}'][\mathtt{k}{:}B \leftarrow \boldsymbol{B}.$
$],\ \vartheta'_1\rangle \xrightarrow{\ell_1} \langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}\overline{B}.\boldsymbol{B}']\varpi_2,\ \vartheta'_2\rangle \xrightarrow{\ell_2} \ldots \xrightarrow{\ell_{n-2}} \langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}\overline{B}.\boldsymbol{B}']$
$\varpi_{n-1},\ \vartheta'_{n-1}\rangle \xrightarrow{\ell_{n-1}} \langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}\overline{B}.\boldsymbol{B}'],\ \vartheta'_n\rangle$ with $\vartheta' \in \vartheta \uparrow mgu(\overline{B}, B)$.

θ' is a valid derivation in $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ because it can be concatenated with the derivations in

$$\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}{:}A \leftarrow \boldsymbol{B}.B\boldsymbol{B}']\vartheta'_n\Theta \quad \text{where} \quad \vartheta'_n = \vartheta_n \uparrow \vartheta$$

which, by the induction hypothesis, are valid derivations in $\mathsf{S}^{\mathsf{t}}[\![P]\!]$, starting with the state $\langle[\dashv\square][\mathtt{i}{:}A \leftarrow \boldsymbol{B}\overline{B}.\boldsymbol{B}'],\ \vartheta'_n\rangle$ which performs a correct junction. ■

Corollary 19 *If all traces in T are derivations of the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ then so are all traces in $\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!]T$.* □

PROOF By (9), all traces in $\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!]T$ have the form

$$\langle [\vdash p(v)],\, \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta} \theta$$

where $\theta \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{.B}]\,\vartheta\,\Theta$. By **Lem. 18**, θ is generated by the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ and starts in state $\langle [\dashv\!\square][\mathtt{i}:A \leftarrow \boldsymbol{.B}],\, \vartheta\rangle$.

If all steps succeed, θ ends in state $\langle [\dashv\!\square][\mathtt{i}:A \leftarrow \boldsymbol{B.}],\, \vartheta'\rangle$ while if instead $\theta \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{.B}]\,\vartheta\,\Theta$ was not successful then θ ends in failure state $\langle [\dashv\!\square]\varpi,\, \vartheta''\rangle$. In both cases, by (2), $\langle [\vdash p(v)],\, \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta}{}^{\mathsf{t}}\, \langle [\dashv\!\square][\mathtt{i}:A \leftarrow \boldsymbol{.B}],\, \vartheta\rangle$ is a valid transition for $\mathsf{S}^{\mathsf{t}}[\![P]\!]$ with $\mathtt{i}:A \leftarrow \boldsymbol{B} \in P$, $\vartheta \in mgu(p(v), A)$, proving that $\langle [\vdash p(v)],\, \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta} \theta$ is a valid trace generated by the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$. ■

The maximal ground derivation semantics of a PROLOG program P can be expressed in fixpoint form for transformer $\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!]$ as follows.

Theorem 20 $\quad \mathsf{S}^{\mathsf{d}}[\![P]\!] = \mathit{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!]$. □

PROOF By continuity of $\hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!]$ and [48], $\mathbf{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!] = \Theta^{\omega}$ where $\Theta^0 \triangleq \varnothing$, $\Theta^{n+1} \triangleq \hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!](\Theta^n)$ and $\Theta^{\omega} \triangleq \bigcup_{n\geq 0} \Theta^n$. We prove the two inclusions separately.

—— All traces in $\Theta^0 = \varnothing$ as well as, by **Cor. 19**, those in Θ^{ω} are derivations of the transition system $\mathsf{S}^{\mathsf{t}}[\![P]\!]$, so we have $\mathbf{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!] \subseteq \mathsf{S}^{\mathsf{d}}[\![P]\!]$.

—— Let $\theta \in \mathsf{S}^{\mathsf{d}}[\![P]\!]$ be a derivation of the transition system. Because all derivations are of the form

$$\theta = \langle [\vdash p(v)],\, \varepsilon\rangle \xrightarrow{\ell_1} \langle \varpi_1,\, \vartheta_1\rangle \xrightarrow{\ell_2} \ldots \xrightarrow{\ell_n} \langle \varpi_n,\, \vartheta_n\rangle$$

we prove that $\theta \in \mathbf{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!]$ by proving that there exists $i \in \mathfrak{N}$ such that $\theta \in \Theta^{i+1}$. We prove that

$$\theta' = \langle [\dashv\!\square][\mathtt{i}:A \leftarrow \boldsymbol{.B}],\, \vartheta\rangle \xrightarrow{\ell_2} \ldots \xrightarrow{\ell_n} \langle \varpi_n,\, \vartheta_n\rangle \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{.B}]\,\vartheta\,\Theta^i$$

for some $i \in \mathfrak{N}$ and so, by (9), $\theta = \langle [\vdash p(v)],\, \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta} \theta' \in \hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!](\Theta^i) = \Theta^{i+1}$.

—— If $\boldsymbol{B} = \boldsymbol{\varepsilon}$ is empty then θ' is reduced to

$$\theta' = \langle [\dashv\!\square][\mathtt{i}:A \leftarrow \boldsymbol{.}],\, \vartheta\rangle \xrightarrow{\mathtt{i}:A\leftarrow|\!\rangle} \langle [\dashv\!\square],\, \vartheta\rangle$$

which, by **Lem. 17**, belongs to $\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{.}]\,\vartheta\,\Theta^i$ for all $i \geqslant 0$.

— If $\boldsymbol{B}$ is not empty, then the proof is by recurrence on the maximal height

$$h = \max\{|\varpi_1|, \ldots, |\varpi_n|\} \geqslant 2$$

of stacks in θ.

— For $h = 2$, $\boldsymbol{B}$ has to be empty $\boldsymbol{\varepsilon}$ which boils down to the previous case.

— If $h > 2$, we consider $\boldsymbol{B} = \boldsymbol{B'B''}$ and we solve the more general problem

$$\langle [\dashv\square][\mathtt{i} : A \leftarrow \boldsymbol{B'.B''}],\, \vartheta_k \rangle \xrightarrow{\ell_{k+1}} \ldots \xrightarrow{\ell_n} \langle \varpi_n,\, \vartheta_n \rangle \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i} : A \leftarrow \boldsymbol{B'.B''}]\, \vartheta_k\, \Theta^i$$

for some $i \in \mathfrak{N}$. The result will follow by considering $\boldsymbol{B'} = \boldsymbol{\varepsilon}$ empty and $\boldsymbol{B} = \boldsymbol{B''}$.

– If $|\boldsymbol{B''}| = 0$ so $\boldsymbol{B''} = \boldsymbol{\varepsilon}$ then

$$\langle [\dashv\square][\mathtt{i} : A \leftarrow \boldsymbol{B'.}],\, \vartheta_k \rangle \xrightarrow{\mathtt{i} : A \leftarrow \boldsymbol{B'} \Vert} \langle [\dashv\square],\, \vartheta_k \rangle \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i} : A \leftarrow \boldsymbol{B'.}]\, \vartheta_k\, \Theta^i$$

for all $i \geqslant 0$.

– Otherwise $|\boldsymbol{B''}| > 0$ so $\boldsymbol{B''} = B\boldsymbol{B'''}$ and $B = p(T)$ for some predicate symbol $p \in \mathbb{p}$ and term $T \in \mathbb{t}$. Then we have

$$\begin{aligned}\langle [\dashv\square][\mathtt{i} : A \leftarrow \boldsymbol{B'.}B\boldsymbol{B'''}],\, \vartheta_k \rangle &\xrightarrow{(\!| \ell : \widetilde{A} \leftarrow \widetilde{\boldsymbol{B}}/\sigma} \langle [\dashv\square][\mathtt{i} : A \leftarrow \boldsymbol{B'}B\boldsymbol{.B'''}][\ell : \widetilde{A} \leftarrow \boldsymbol{.}\widetilde{\boldsymbol{B}}],\, \vartheta_{k+1} \rangle \\ &\xrightarrow{\ell_{k+2}} \ldots \xrightarrow{\ell_n} \langle \varpi_n,\, \vartheta_n \rangle\end{aligned}$$

where $\sigma = mgu(\vartheta_k(B), \widetilde{A})$ and $\vartheta_{k+1} = \vartheta_k \uparrow \sigma$.

By **Lem. 6**, $\alpha^p(\theta) \in \mathbb{D}_{\mathscr{P},\varnothing}$ so θ is well-parenthesized, i.e. either $\langle \varpi_n,\, \vartheta_n \rangle$ is a failure state including $[\ell : \widetilde{A} \leftarrow \boldsymbol{.}\widetilde{\boldsymbol{B}}]$ in the stack or we must have $m < n$ such that

$$\langle \varpi_m,\, \vartheta_m \rangle = \langle \varpi[\ell : \widetilde{A} \leftarrow \boldsymbol{.}\widetilde{\boldsymbol{B}}],\, \vartheta_m \rangle \xrightarrow{\ell : \widetilde{A} \leftarrow \widetilde{\boldsymbol{B}} \Vert} \langle \varpi,\, \vartheta_m \rangle$$

with $\varpi = \varpi_{m+1}$ and $\vartheta_m = \vartheta_{m+1}$. Observe that in θ':

- The height of the stack is increased by 1 in (3) and decreased by 1 in (4).

- In case of success, θ' is well-parenthesized and so the stack has the same height on matching parentheses. Moreover the transition never changes the bottom of the stack. Because

$$\langle \underbrace{[\dashv\square][\mathtt{i} : A \leftarrow \boldsymbol{B'.}B\boldsymbol{B'''}]}_{\varpi_k},\, \vartheta_k \rangle \xrightarrow{(\!| \ell : \widetilde{A} \leftarrow \widetilde{\boldsymbol{B}}/\sigma} \langle \underbrace{[\dashv\square][\mathtt{i} : A \leftarrow \boldsymbol{B'}B\boldsymbol{.B'''}][\ell : \widetilde{A} \leftarrow \boldsymbol{.}\widetilde{\boldsymbol{B}}]}_{\varpi_{k+1}},\, \vartheta_{k+1} \rangle$$

and symmetrically

$$\langle \underbrace{[\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}''']}_{\varpi_m}[\ell:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}\boldsymbol{.}],\ \vartheta_m\rangle \xrightarrow{\ell:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}\,\rangle\!\rangle} \langle \underbrace{[\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}''']}_{\varpi_{m+1}},\ \vartheta_{m+1}\rangle$$

we can write the trace from k to $m+1$ as

$$\langle [\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'\boldsymbol{.}B\boldsymbol{B}'''],\ [\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}'''],\ \vartheta_k\rangle \Uparrow^{\mathsf{d}} \theta''$$

where

$$\theta'' \triangleq \langle [\vdash p(v)],\varepsilon\rangle \xrightarrow{\langle\!\langle\, \ell:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}/\sigma} \langle [\dashv\Box][\ell:\widetilde{A}\leftarrow\boldsymbol{.}\widetilde{\boldsymbol{B}}],\sigma'\rangle \xrightarrow{\ell_{k+1}} \ldots \xrightarrow{\ell_m} \langle [\dashv\Box][\ell:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}\boldsymbol{.}],\vartheta'_m\rangle$$

where p is the predicate symbol of $\widetilde{A}$ and $\sigma' = mgu(p(v),\widetilde{A})$. Note that by (4) we have

$$\langle [\dashv\Box][\ell:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}\boldsymbol{.}],\ \vartheta'_m\rangle \xrightarrow{\ell:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}\,\rangle\!\rangle^{\mathsf{t}}} \langle [\dashv\Box],\ \vartheta'_m\rangle$$

where $\mathtt{i}:\widetilde{A}\leftarrow\widetilde{\boldsymbol{B}}\in P$ and $\vartheta'_{m+1}=\vartheta'_m$. Since the maximal length of stacks in θ'' is strictly less that in θ', there exists, by induction hypothesis, a Θ^q such that $\theta''\in\Theta^q.p(v)$. Therefore, by monotonicity

$$\langle [\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'\boldsymbol{.}B\boldsymbol{B}'''],\ [\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}'''],\ \vartheta_k\rangle \Uparrow^{\mathsf{d}} \theta'' \in \Theta^t.p(v)$$

for all $t\geqslant q$.

Let us define

$$\theta''' \triangleq \langle [\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}'''],\ \vartheta_{m+1}\rangle \xrightarrow{\ell_{m+2}} \ldots \xrightarrow{\ell_n} \langle \varpi_n,\ \vartheta_n\rangle\ .$$

Because $|\boldsymbol{B}'''|<|\boldsymbol{B}''|$, there exists, by induction, some $j\geqslant 0$ such that

$$\theta'''\in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}''']\,\vartheta_{m+1}\,\Theta^j$$

By the increasing fixpoint computation of the chain $\{T^i\}_{i\geqslant 0}$ and $\hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}$ monotonicity, we have

$$\theta'''\in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{.B}''']\,\vartheta_{m+1}\,\Theta^t \quad \text{for all } t\geqslant j.$$

By letting $t=\max\{j,q\}$, the theorem follows. ∎

Example 21 For the PROLOG program P of **Ex. 1**, the fixpoint equation (9) is $\Theta=\hat{\mathsf{F}}^{\mathsf{d}}[\![\overline{P}]\!](\Theta)$ of the form

$$\begin{array}{l}\Theta = \{\langle[\vdash \mathtt{n}(x_0)],\ \varepsilon\rangle \xrightarrow{\langle\!\langle\, \mathtt{0}:\mathtt{n}(0)\leftarrow/\{x_0\leftarrow\mathtt{0}\}} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{0}:\mathtt{n}(\mathtt{0})\leftarrow\boldsymbol{.}]\,\{x_0\leftarrow\mathtt{0}\}\,\Theta\}\cup\\ \quad \{\langle[\vdash \mathtt{n}(x_1)],\ \varepsilon\rangle \xrightarrow{\langle\!\langle\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)/\{x_1\leftarrow\mathtt{s}(x_2)\}} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\ \leftarrow\ \boldsymbol{.}\mathtt{n}(x_2)]\\ \quad \{x_1\leftarrow\mathtt{s}(x_2)\}\,\Theta\}\end{array}$$

$$= \{\langle[\vdash \mathtt{n}(x_0)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow/\{x_0\leftarrow 0\}} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare]\,\{x_0 \leftarrow 0\}\,\Theta\} \cup$$
$$\{\langle[\vdash \mathtt{n}(x_1)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)/\{x_1\leftarrow\mathtt{s}(x_2)\}} \{(\langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \blacksquare\mathtt{n}(x_2)],$$
$$[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare],\ \{x_1 \leftarrow \mathtt{s}(x_2)\}\rangle \Uparrow^{\mathsf{d}} \eta \xrightarrow{\ell} \langle\varpi,\ \vartheta'\rangle) \fatsemi \theta \mid \eta \xrightarrow{\ell} \langle\varpi,$$
$$\vartheta'\rangle \in \Theta.B' \wedge \sigma \in mgu(\mathtt{n}(x_2), B') \wedge \theta \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare]\,(\{x_1 \leftarrow$$
$$\mathtt{s}(x_2)\} \uparrow \sigma \uparrow \vartheta')\,\Theta\}\}$$
$$= \{\langle[\vdash \mathtt{n}(x_0)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow/\{x_0\leftarrow 0\}} \langle[\dashv\square][\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_0 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle}$$
$$\langle[\dashv\square],\ \{x_0 \leftarrow 0\}\rangle\} \cup$$
$$\{\langle[\vdash \mathtt{n}(x_1)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)/\{x_1\leftarrow\mathtt{s}(x_2)\}} \{(\langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \blacksquare\mathtt{n}(x_2)],$$
$$[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare],\ \{x_1 \leftarrow \mathtt{s}(x_2)\}\rangle \Uparrow^{\mathsf{d}} \eta \xrightarrow{\ell} \langle\varpi, \vartheta'\rangle) \fatsemi \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))$$
$$\leftarrow \mathtt{n}(x_2)\blacksquare],\ (\{x_1 \leftarrow \mathtt{s}(x_2)\} \uparrow \sigma \uparrow \vartheta')\rangle \xrightarrow{\mathtt{i}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)\,|\!\rangle} \langle[\dashv\square],\ (\{x_1 \leftarrow$$
$$\mathtt{s}(x_2)\} \uparrow \sigma \uparrow \vartheta')\rangle \mid \eta \xrightarrow{\ell} \langle\varpi,\ \vartheta'\rangle \in \Theta.B' \wedge \sigma \in mgu(\mathtt{n}(x_2), B')\}\}$$

The first iterates of the fixpoint computation for finite traces are as follows:

$$\Theta_0 = \varnothing$$

$$\Theta_1 = \{\langle[\vdash \mathtt{n}(x_0)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow} \langle[\dashv\square][\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_0 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle} \langle[\dashv\square],$$
$$\{x_0 \leftarrow 0\}\rangle\}$$

$$\Theta_2 = \{\langle[\vdash \mathtt{n}(x_0)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow} \langle[\dashv\square][\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_0 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle} \langle[\dashv\square],$$
$$\{x_0 \leftarrow 0\}\rangle,$$
$$\langle[\vdash \mathtt{n}(x_1)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \blacksquare\mathtt{n}(x_2)],\ \{x_1 \leftarrow$$
$$\mathtt{s}(x_2)\}\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare][\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_1 \leftarrow$$
$$\mathtt{s}(x_2), x_2 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare],\ \{x_1 \leftarrow \mathtt{s}(x_2), x_2 \leftarrow$$
$$0\}\rangle \xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)\,|\!\rangle} \langle[\dashv\square],\ \{x_1 \leftarrow \mathtt{s}(x_2), x_2 \leftarrow 0\}\rangle\}$$

$$\Theta_3 = \{\langle[\vdash \mathtt{n}(x_0)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow} \langle[\dashv\square][\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_0 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle} \langle[\dashv\square],$$
$$\{x_0 \leftarrow 0\}\rangle,$$
$$\langle[\vdash \mathtt{n}(x_1)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \blacksquare\mathtt{n}(x_2)],\ \{x_1 \leftarrow$$
$$\mathtt{s}(x_2)\}\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare][\mathtt{0}:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_1 \leftarrow$$
$$\mathtt{s}(x_2), x_2 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle} \langle[\dashv\square][\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2)) \leftarrow \mathtt{n}(x_2)\blacksquare],\ \{x_1 \leftarrow \mathtt{s}(x_2), x_2 \leftarrow$$
$$0\}\rangle \xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x_2))\leftarrow\mathtt{n}(x_2)\,|\!\rangle} \langle[\dashv\square],\ \{x_1 \leftarrow \mathtt{s}(x_2), x_2 \leftarrow 0\}\rangle,$$
$$\langle[\vdash \mathtt{n}(x_3)],\ \varepsilon\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_4))\leftarrow\mathtt{n}(x_4)} \langle[\dashv\square][:\mathtt{n}(\mathtt{s}(x_4)) \leftarrow \blacksquare\mathtt{n}(x_4)],\ \{x_3 \leftarrow$$
$$\mathtt{s}(x_4)\}\rangle \xrightarrow{\langle\!|\, \mathtt{1}:\mathtt{n}(\mathtt{s}(x_5))\leftarrow\mathtt{n}(x_5)} \langle[\dashv\square][:\mathtt{n}(\mathtt{s}(x_4)) \leftarrow \mathtt{n}(x_4)\blacksquare][:\mathtt{n}(\mathtt{s}(x_5)) \leftarrow \blacksquare\mathtt{n}(x_5)],$$
$$\{x_3 \leftarrow \mathtt{s}(x_4), x_4 \leftarrow \mathtt{s}(x_5)\}\rangle \xrightarrow{\langle\!|\, \mathtt{0}:\mathtt{n}(0)\leftarrow} \langle[\dashv\square][:\mathtt{n}(\mathtt{s}(x_4)) \leftarrow \mathtt{n}(x_4)\blacksquare][:\mathtt{n}(\mathtt{s}(x_5))$$
$$\leftarrow \blacksquare\mathtt{n}(x_5)][:\mathtt{n}(0) \leftarrow \blacksquare],\ \{x_3 \leftarrow \mathtt{s}(x_4), x_4 \leftarrow \mathtt{s}(x_5), x_5 \leftarrow 0\}\rangle \xrightarrow{\mathtt{0}:\mathtt{n}(0)\leftarrow|\!\rangle}$$
$$\langle[\dashv\square][:\mathtt{n}(\mathtt{s}(x_4)) \leftarrow \mathtt{n}(x_4)\blacksquare][:\mathtt{n}(\mathtt{s}(x_5)) \leftarrow \blacksquare\mathtt{n}(x_5)],\ \{x_3 \leftarrow \mathtt{s}(x_4), x_4 \leftarrow$$
$$\mathtt{s}(x_5), x_5 \leftarrow 0\}\rangle \xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x_5))\leftarrow\mathtt{n}(x_5)\,|\!\rangle} \langle[\dashv\square][:\mathtt{n}(\mathtt{s}(x_4)) \leftarrow \mathtt{n}(x_4)\blacksquare],\ \{x_3 \leftarrow$$
$$\mathtt{s}(x_4), x_4 \leftarrow \mathtt{s}(x_5), x_5 \leftarrow 0\}\rangle \xrightarrow{\mathtt{1}:\mathtt{n}(\mathtt{s}(x_4))\leftarrow\mathtt{n}(x_4)\,|\!\rangle} \langle[\dashv\square],\ \{x_3 \leftarrow \mathtt{s}(x_4), x_4 \leftarrow$$
$$\mathtt{s}(x_5), x_5 \leftarrow 0\}\rangle\}$$

$\Theta_4 = \ldots$ □

12.3 Fixpoint s-semantics

Let us define the *bottom-up call-patterns transformer* $\hat{\mathsf{F}}^{\mathsf{s}}[\![P]\!] \in \wp(\mathbb{A}) \mapsto \wp(\mathbb{A})$ for a PROLOG program $P \in \mathbb{P}$ as

$$\hat{\mathsf{F}}^{\mathsf{s}}[\![P]\!] \triangleq \boldsymbol{\lambda}\mathscr{A} \cdot \bigcup_{\mathtt{i}:A \leftarrow \boldsymbol{B} \in P} \{\vartheta(A) \mid \vartheta \in \hat{\mathsf{F}}^{\mathsf{s}}_{\cdot}[\mathtt{i}:A \leftarrow \boldsymbol{.B}]\,\mathscr{A}\,\{\varepsilon\}\} \tag{12}$$

where the *clause transformer* $\hat{\mathsf{F}}^{\mathsf{s}}_{\cdot}[\mathtt{i}:A \leftarrow \boldsymbol{B.B'}] \in \wp(\boldsymbol{\Theta}) \mapsto \wp(\mathbb{S}) \mapsto \wp(\mathbb{S})$ is defined as

$$\hat{\mathsf{F}}^{\mathsf{s}}_{\cdot}[\mathtt{i}:A \leftarrow \boldsymbol{B.}B\boldsymbol{B'}] \triangleq \boldsymbol{\lambda}\mathscr{A} \cdot \boldsymbol{\lambda}\mathscr{S} \cdot \{\vartheta' \mid B' \in \mathscr{A} \wedge \sigma \in mgu(B,B') \wedge \vartheta \in \mathscr{S} \wedge \vartheta' \in \hat{\mathsf{F}}^{\mathsf{s}}_{\cdot}[\mathtt{i}:A \leftarrow \boldsymbol{B}B\boldsymbol{.B'}]\,\mathscr{A}\,(\vartheta \uparrow \sigma)\} \tag{13}$$

$$\hat{\mathsf{F}}^{\mathsf{s}}_{\cdot}[\mathtt{i}:A \leftarrow \boldsymbol{B.}] \triangleq \boldsymbol{\lambda}\mathscr{A} \cdot \boldsymbol{\lambda}\mathscr{S} \cdot \mathscr{S}\ . \tag{14}$$

Example 22 For the PROLOG program P of **Ex. 1**, the fixpoint equation (12) of the form $\mathscr{A} = \hat{\mathsf{F}}^{\mathsf{sd}}[\![P]\!](\mathscr{A})$ is

$$\mathscr{A} = \{\mathtt{n}(\mathtt{0})\} \cup \{\sigma(\mathtt{n}(\mathtt{s}(x))) \mid B' \in \mathscr{A} \wedge \sigma \in mgu(\mathtt{n}(x), B')\}$$

The iterates of the fixpoint computation for finite traces are as follows

$$\begin{aligned}
\mathscr{A}^0 &= \varnothing \\
\mathscr{A}^1 &= \{\mathtt{n}(\mathtt{0})\} \\
\mathscr{A}^2 &= \{\mathtt{n}(\mathtt{0}), \mathtt{n}(\mathtt{s}(\mathtt{0}))\} \\
&\ldots \\
\mathscr{A}^k &= \{\mathtt{n}(\mathtt{s}^i(\mathtt{0})) \mid i = 0, ..., k-1\} && \text{induction hypothesis} \\
\mathscr{A}^{k+1} &= \{\mathtt{n}(\mathtt{0})\} \cup \{\sigma(\mathtt{n}(\mathtt{s}(x))) \mid B' \in \mathscr{A}^k \wedge \sigma \in mgu(\mathtt{n}(x), B')\} \\
&= \{\mathtt{n}(\mathtt{0})\} \cup \{\sigma(\mathtt{n}(\mathtt{s}(x))) \mid \sigma \in mgu(\mathtt{n}(x), \mathtt{n}(\mathtt{s}^i(\mathtt{0}))) \wedge i = 1, ..., k-1\} \\
&= \{\mathtt{n}(\mathtt{0})\} \cup \{\mathtt{n}(\mathtt{s}(\mathtt{s}^i(\mathtt{0}))) \mid i = 1, ..., k-1\} \\
&= \{\mathtt{n}(\mathtt{s}^i(\mathtt{0})) \mid i = 0, ..., k\} \\
&\ldots \\
\mathscr{A}^\omega &= \bigcup_{k \geq 0} \mathscr{A}^k = \{\mathtt{n}(\mathtt{s}^i(\mathtt{0})) \mid i \geq 0\} && \text{limit}
\end{aligned}$$ □

Lemma 23 $\alpha^{\mathsf{s}} \circ \hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!] = \hat{\mathsf{F}}^{\mathsf{s}}[\![P]\!] \circ \alpha^{\mathsf{s}}$ *where* $\alpha^{\mathsf{s}} \triangleq \alpha^{\mathsf{m}} \circ \alpha^{\mathsf{p}} \circ \alpha^{\mathsf{K}} \circ \alpha^{\mathsf{sd}}$. □

PROOF We must prove that

$$\alpha^{\mathsf{s}}(\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!](\Theta)) = \hat{\mathsf{F}}^{\mathsf{s}}[\![P]\!](\alpha^{\mathsf{s}}(\Theta))$$

where by (9) and α^{s} is a complete join morphism, we have

$$\alpha^{\mathsf{s}}(\hat{\mathsf{F}}^{\mathsf{d}}[\![P]\!])\Theta = \bigcup_{\mathtt{i}:A\leftarrow \boldsymbol{B}\notin P,\, p\in\mathbb{P},\, v\in\mathbb{V},\, \vartheta\in mgu(p(v),A)} \alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \varepsilon\rangle \xrightarrow{(\!|\mathtt{i}:A\leftarrow \boldsymbol{B}/\vartheta} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{.B}]\,\vartheta\,\Theta)$$

and by (12)

$$\hat{\mathsf{F}}^{\mathsf{s}}[\![P]\!](\alpha^{\mathsf{s}}(\Theta)) = \bigcup_{\mathtt{i}:A\leftarrow \boldsymbol{B}\notin P} \{\vartheta(A) \mid \vartheta \in \hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{.B}]\,(\alpha^{\mathsf{s}}(\Theta))\,\{\varepsilon\}\}$$

so we have to prove that for all $\mathtt{i}:A\leftarrow \boldsymbol{B}\notin P$, we have

$$\begin{array}{ll} & \alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \sigma\rangle \xrightarrow{(\!|\mathtt{i}:A\leftarrow \boldsymbol{B'B}/\vartheta} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.B}]\,(\vartheta\uparrow\sigma)\,\Theta) \\ = & \{\vartheta'(A) \mid \vartheta' \in \hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.B}]\,(\alpha^{\mathsf{s}}(\Theta))\,\{\sigma\}\} \qquad \text{where} \quad \vartheta\in mgu(p(v),A) \end{array}$$

We proceed by induction on the length of $\boldsymbol{B}$

- For the base $\boldsymbol{B}=\boldsymbol{\varepsilon}$, we have

$$\begin{array}{ll} & \alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \sigma\rangle \xrightarrow{(\!|\mathtt{i}:A\leftarrow /(\vartheta\uparrow\sigma)} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.}]\,(\vartheta\uparrow\sigma)\,\Theta) \\ = & \quad \wr\text{by def. (11) of } \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.}](\vartheta\uparrow\sigma)\Theta\wr \\ & \alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \sigma\rangle \xrightarrow{(\!|\mathtt{i}:A\leftarrow /(\vartheta\uparrow\sigma)} \{\langle[\dashv\Box][\mathtt{i}:A\leftarrow \boldsymbol{B'.}],\, (\vartheta\uparrow\sigma)\rangle \xrightarrow{\mathtt{i}:A\leftarrow |\!)} \langle[\dashv\Box],\, (\vartheta\uparrow\sigma)\rangle\}) \\ = & \quad \wr\text{by def. } \alpha^{\mathsf{s}}\wr \\ & \{(\vartheta\uparrow\sigma)(p(v))\} \\ = & \{\sigma(A)\} \qquad\qquad \wr\text{since } \vartheta\in mgu(p(v),A)\wr \\ = & \{\vartheta'(A) \mid \vartheta'\in\{\sigma\}\} \\ = & \quad \wr\hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.}]\alpha^{\mathsf{s}}(\Theta)\{\sigma\}=\{\sigma\} \text{ by } (14)\wr \\ & \{\vartheta'(A) \mid (\vartheta'\uparrow\sigma)\in \hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.}]\alpha^{\mathsf{s}}(\Theta)\{\sigma\}\}\ . \end{array}$$

- For the induction step $\boldsymbol{B}=B\boldsymbol{B''}$, we have

$$\begin{array}{ll} & \alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \sigma\rangle \xrightarrow{(\!|\mathtt{i}:A\leftarrow \boldsymbol{B'}B\boldsymbol{B''}/\vartheta} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.}B\boldsymbol{B''}]\,\vartheta\,\Theta) \\ = & \quad \wr\text{by def (10) of } \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A\leftarrow \boldsymbol{B'.}B\boldsymbol{B''}]\vartheta\Theta\wr \end{array}$$

$\alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \sigma\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{B}''/\vartheta} \{(\langle[\dashv\square][\mathtt{i}:A \leftarrow \boldsymbol{B}'.B\boldsymbol{B}''],\, [\dashv\square][\mathtt{i}:A \leftarrow B.\boldsymbol{B}''],\, \vartheta\rangle \Uparrow^{\mathsf{d}} \eta \xrightarrow{\ell} \langle\varpi,\, \vartheta'\rangle) \fatsemi \theta \mid \eta \xrightarrow{\ell} \langle\varpi,\, \vartheta'\rangle \in \Theta. B' \wedge \sigma \in mgu(B, B') \wedge \theta \in \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A \leftarrow B.\boldsymbol{B}''] \,(\vartheta \uparrow \sigma \uparrow \vartheta')\, \Theta\}) \triangleq \mathcal{X}$

We have $H \in \mathcal{X}$ if and only if there exists a σ' such that $H = \sigma'(p(v))$ and $\langle[\vdash p(v)],\, \sigma\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{B}''/\vartheta} (\langle[\dashv\square][\mathtt{i}:A \leftarrow \boldsymbol{B}'.B\boldsymbol{B}''],\, [\dashv\square][\mathtt{i}:A \leftarrow B.\boldsymbol{B}''],\, \vartheta\rangle \Uparrow^{\mathsf{d}} \eta \xrightarrow{\ell} \langle\varpi,\, \vartheta'\rangle)$ is successful for B' so that $\varpi = [\dashv\square]$, $\vartheta'(B) \in \alpha^{\mathsf{s}}(\Theta)$, and, by definition of inlaying in **Sec. 12.1**, $\sigma' = \sigma \uparrow \vartheta \uparrow \vartheta'$.

By induction hypothesis for $\boldsymbol{B}''$, $\alpha^{\mathsf{s}}(\langle[\vdash p(v)],\, \sigma \uparrow \vartheta'\rangle \xrightarrow{\langle\!|\, \mathtt{i}:A\leftarrow \boldsymbol{B}'B\boldsymbol{B}''/\vartheta} \hat{\mathsf{F}}^{\mathsf{d}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{B}'B.\boldsymbol{B}''] \,\vartheta\, \Theta) = \{\vartheta''(A) \mid \vartheta'' \in \hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{B}'B.\boldsymbol{B}''] \,(\alpha^{\mathsf{s}}(\Theta))\, \{\sigma \uparrow \vartheta'\}\}$ and so $\sigma' = \sigma \uparrow \vartheta \uparrow \vartheta' \in \hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{B}'.B\boldsymbol{B}''] \,(\alpha^{\mathsf{s}}(\Theta))\, \{\sigma\}$ proving that

$$H \in \{\vartheta'''(A) \mid \vartheta''' \in \hat{\mathsf{F}}^{\mathsf{s}}_{\bullet}[\mathtt{i}:A \leftarrow \boldsymbol{B}'.B\boldsymbol{B}''] \,(\alpha^{\mathsf{s}}(\Theta))\, \{\sigma\}\} \;. \qquad \blacksquare$$

The fixpoint s-semantics of [46] is an abstract interpretation of the fixpoint bottom-up most general maximal derivation semantics of **Sec. 12.2**.

Theorem 24 (G. Levi et al.) $\quad \mathsf{S}^{\mathsf{s}}[\![P]\!] = \mathit{lfp}^{\subseteq} \hat{\mathsf{F}}^{\mathsf{s}}[\![P]\!]$. $\square$

Proof By **Lem. 23** and [49, Th. 7.1.0.4 (3)]. $\blacksquare$

13 Conclusion

We showed how abstract interpretation of the maximal trace semantics of a simple grammar-based language, akin the semantics of context-free grammars and pushdown automata [35], can provide a comprehensive view of most well-known semantics of resolution-based languages such as logic programming and Prolog. Other semantics can be derived similarly, for instance for modelling infinite computations by combining inductive and co-inductive semantics [50] and for modelling different forms of negation [27]. The result is a uniform specification framework for interpreters of logic programs which can be systematically designed by consecutive abstractions of a basic abstract machine. The analogy with the semantics of grammars is, in this context, striking. We believe that both the semantics of grammars and that of resolution-based languages can be specified in a uniform way as instances of a unique transition system semantics involving more expressive grammars inspired by Prolog, rewriting, etc. Having formalized logic program semantics by abstract interpretation, may provide the way to integrate its correctness proofs with that of its decidable abstractions, such as those for static analysis. Because abstraction can be constructed by calculational design [51], as shown in the formal

proofs, a proof assistant or theorem prover can be used to automatically check or perform these calculations. This leads to formally verified implementations and static analyzers.

References

[1] M. van Emden, R. Kowalski, The semantics of predicate logic as a programming language, Journal of the ACM 23 (4) (1976) 733–742.

[2] T. Przymusinski, On the declarative and procedural semantics of logic programs, Journal of Automated Reasoning 5 (2) (1989) 201–228.

[3] N. D. Jones, A. Mycroft, Stepwise development of operational and denotational semantics for Prolog, in: S.-Å. Tärnlund (Ed.), Proc. 2nd Int'l Conf. on Logic Programming, IEEE Computer Society Press, 1984, pp. 281–288.

[4] S. K. Debray, P. Mishra, Denotational and operational semantics for Prolog, in: M. Wirsing (Ed.), Formal Description of Programming Concepts III, North-Holland, 1987, pp. 245–269.

[5] N. D. Jones, H. Søndergaard, A semantics-based framework for the abstract interpretation of Prolog, in: S. Abramsky, C. Hankin (Eds.), Abstract Interpretation of Declarative Languages, Ellis Horwood Ltd, 1987, pp. 123–142.

[6] B. Arbab, D. M. Berry, Operational and denotational semantics of Prolog, Journal of Logic Programming 4 (4) (1987) 309–330.

[7] A. de Bruin, E. de Vink, Continuation semantics for Prolog with cut, in: J. Diaz, F. Orejas (Eds.), Proc. CAAP 89, Vol. 351 of Lecture Notes in Computer Science, Springer, 1989, pp. 178–192.

[8] E. Börger, A logical operational semantics of full Prolog, in: E.Börger, H. Kleine, H. Büning, M. Richter (Eds.), CSL 89. 3rd workshop on Computer Science Logic, Vol. 440 of Lecture Notes in Computer Science, Springer, 1990, pp. 36–64.

[9] R. Barbuti, M. Codish, R. Giacobazzi, G. Levi, Modelling Prolog control, in: Proc. 19th Annual ACM Symp. on Principles of Programming Languages, ACM Press, 1992, pp. 95–104.

[10] R. Barbuti, M. Codish, R. Giacobazzi, M. Maher, Oracle semantics for Prolog, Information and Computation 122 (2) (1995) 178–200.

[11] M. J. Maher, Equivalences of logic programs, in: J. Minker (Ed.), Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann, 1988, pp. 627–658.

[12] G. Levi, Models, unfolding rules and fixpoint semantics, in: R. A. Kowalski, K. A. Bowen (Eds.), Proc. 5th Int'l Conf. on Logic Programming, The MIT Press, 1988, pp. 1649–1665.

[13] M. J. Maher, R. Ramakrishnan, Déjà vu in fixpoints of logic programs, in: E. Lusk, R. Overbeek (Eds.), Proc. North American Conf. on Logic Programming'89, The MIT Press, 1989, pp. 963–980.

[14] M. Falaschi, G. Levi, M. Martelli, C. Palamidessi, Declarative modeling of the operational behavior of logic languages, Theoretical Computer Science 69 (3) (1989) 289–318.

[15] M. Falaschi, G. Levi, M. Martelli, C. Palamidessi, A model-theoretic reconstruction of the operational semantics of logic programs, Information and Computation 102 (1) (1993) 86–113.

[16] F. Denis, J.-P. Delahaye, Unfolding, procedural and fixpoint semantics of logic programs, in: C. Choffrut, M. Jantzen (Eds.), Proc. 8th Int'l Symp. on Theoretical Aspects of Computer Science (*STACS '91*), Vol. 480 of Lecture Notes in Computer Science, Springer, 1991, pp. 511–522.

[17] M. Gabbrielli, M. C. Meo, Fixpoint semantics for partial computed answer substitutions and call patterns, in: H. Kirchner, G. Levi (Eds.), Algebraic and Logic Programming, Proc. Third Int'l Conf., Vol. 632 of Lecture Notes in Computer Science, Springer, 1992, pp. 84–99.

[18] M. Gabbrielli, G. Levi, M. C. Meo, Observable behaviors and equivalences of logic programs, Information and Computation 122 (1) (1996) 1–29.

[19] M. Gabbrielli, G. Levi, M. C. Meo, Resultant semantics for Prolog, Journal of Logic and Computation 6 (4) (1996) 491–521.

[20] H. Gaifman, E. Shapiro, Fully abstract compositional semantics for logic programs, in: Conf. Record of the 16th ACM Symp. on Principles of Programming Languages (*POPL '89*), ACM Press, 1989, pp. 134–142.

[21] A. Bossi, M. Gabbrielli, G. Levi, M. C. Meo, A compositional semantics for logic programs, Theoretical Computer Science 122 (1–2) (1994) 3–47.

[22] A. Bossi, M. Gabbrielli, G. Levi, M. C. Meo, An OR-compositional semantics for logic programs, in: Constructing logic programs, John Wiley & Sons, Inc., 1993, pp. 215–240.

[23] A. Bossi, M. Gabbrielli, G. Levi, M. C. Meo, A compositional semantics for logic programs, Theoretical Computer Science 122 (1–2) (1994) 3–47.

[24] A. Brogi, F. Turini, Fully abstract compositional semantics for an algebra of logic programs, Theoretical Computer Science 149 (2) (1995) 201–229.

[25] M. Comini, M. C. Meo, Compositionality properties of *SLD*-derivations, Theoretical Computer Science.

[26] A. Bossi, M. Gabbrielli, G. Levi, M. Martelli, The s-semantics approach: theory and applications, Journal of Logic Programming 19-20 (1994) 149–197.

[27] K. R. Apt, Introduction to logic programming, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics, Elsevier and MIT Press, 1990, pp. 495–574.

[28] J. Lloyd, Foundations of Logic Programming, 2nd Edition, Springer, 1987.

[29] L. Sterling, E. Shapiro, The Art of Prolog — Advanced Programming Techniques, 2nd Edition, The MIT Press, 1994.

[30] R. Giacobazzi, "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics, in: C. Puech (Ed.), Proc. 13th Int'l Symp. on Theoretical Aspects of Computer Science (*STACS '96*), Vol. 1046 of Lecture Notes in Computer Science, Springer, 1996, pp. 503–514.

[31] R. Giacobazzi, F. Ranzato, Complementing logic program semantics, in: M. Hanus, M. Rodríguez Artalejo (Eds.), Proc. 5th Int'l Conf. on Algebraic and Logic Programming (ALP '96), Vol. 1139 of Lecture Notes in Computer Science, Springer, 1996, pp. 238–253.

[32] R. Giacobazzi, F. Ranzato, Uniform closures: Order-theoretically reconstructing logic program semantics and abstract domain refinements, Information and Computation 145 (2) (1998) 153–190.

[33] R. Giacobazzi, F. Ranzato, The reduced relative power operation on abstract domains, Theoretical Computer Science 216 (1-2) (1999) 159–211.

[34] C. Palamidessi:, Algebraic properties of idempotent substitutions, in: M. Paterson (Ed.), Proc. 17th Int'l Colloquium on Automata, Languages and Programming '97, Vol. 443 of Warwick University, England, Lecture Notes in Computer Science, Springer, 1990, pp. 386–399.

[35] P. Cousot, R. Cousot, Grammar analysis and parsing by abstract interpretation, in: T. Reps, M. Sagiv, J. Bauer (Eds.), Program Analysis and Compilation, Theory and Practice: Essays dedicated to Reinhard Wilhelm, Vol. 4444 of Lecture Notes in Computer Science, Springer, 2006, pp. 178–203.

[36] D. Warren, Implementing Prolog - compiling predicate logic programs, DAI research reports 39–40, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland (1977).

[37] H. Aït-Kaci, Warren's Abstract Machine: A Tutorial Reconstruction, The MIT Press, 1991.

[38] K. Apt, Logic programming, in: J. Van Leeuwen (Ed.), Formal Models and Semantics, Vol. B of Handbook of Theoretical Computer Science, Elsevier, 1990, Ch. 10, pp. 493–574.

[39] R. Kowalski, Predicate logic as a programming language, in: Proc. Congress IFIP 1974, North-Holland, 1974, pp. 569–574.

[40] A. Colmerauer, H. Kanoui, M. V. Caneghem, Last steps towards an ultimate Prolog, in: P. Hayes (Ed.), Proc. 7th Int'l Joint Conf. on Artificial Intelligence (IJCAI'81), Vancouver, British Columbia, Canada, Vol. 2, William Kaufmann, 1981, pp. 947–948.

[41] A. Colmerauer, P. Roussel, The birth of Prolog, in: History of programming languages, Vol. II, ACM Press, 1996, pp. 331–367.

[42] F. Bueno, D. Cabeza, M. Carro, M. Hermenegildo, P. López-García, G. Puebla, The Ciao Prolog system. Reference manual, Tech. Rep. CLIP3/97.1, School of Computer Science, Technical University of Madrid (UPM), available from `http://www.clip.dia.fi.upm.es/` (August 1997).

[43] M. Gabbrielli, R. Giacobazzi, Goal independency and call patterns in the analysis of logic programs, in: Proc. 9th ACM Symp. on Applied Computing, Phoenix, Arizona, ACM Press, 1994, pp. 394–399.

[44] K. Clarke, Predicate logic as a computational formalism, Research Monograph 79/59, Department of Computing, Imperial College, London, Great Britain (December 1979).

[45] M. Falaschi, G. Levi, M. Martelli, C. Palamidessi, Declarative modelling of the operational behavior of logic languages, Theoretical Computer Science 69 (1989) 289–318.

[46] A. Bossi, M. Gabbrielli, G. Levi, M. Martelli, The s-semantics approach: Theory and applications, Journal of Logic Programming 19–20 (1994) 149–197.

[47] M. Gabbrielli, G. Levi, M. Meo, Observable behaviors and equivalences of logic programs, Information and Computation 122 (1) (1995) 1–29.

[48] A. Tarski, A lattice theoretical fixpoint theorem and its applications, Pacific Journal of Mathematics 5 (1955) 285–310.

[49] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Conf. Record of the 6th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languagesm, San Antonio, Texas, ACM Press, 1979, pp. 269–282.

[50] P. Cousot, Constructive design of a hierarchy of semantics of a transition system by abstract interpretation, Theoretical Computer Science 277 (1—2) (2002) 47–103.

[51] P. Cousot, The calculational design of a generic abstract interpreter, in: M. Broy, R. Steinbrüggen (Eds.), Calculational System Design, Vol. 173, NATO Science Series, Series F: Computer and Systems Sciences. IOS Press, Amsterdam, 1999, pp. 421–505.