(Ao.Univ.Prof. Dr.phil. Matthias Baaz)

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

D I P L O M A R B E I T

On the Expressibility of Semi-Unification Problems

Decidability and Undecidability Results

Ausgeführt am Institut für

Diskrete Mathematik und Geometrie
der Technischen Universität Wien

unter der Anleitung von **Ao.Univ.Prof. Dr.phil. Matthias Baaz**

durch

Florian Pflug
Matrikelnummer: 0026535

Förstergasse 6/35
1020 Wien

Wien, 14. Oktober 2014

(Florian Pflug)

Kurzfassung

Das *Semi-Unifikationsproblem* ist eine naheliegende Verallgemeinerung des wohlbekannten *Unifikationsproblems*, und besitzt Anwendungen in verschiedenen Feldern der theoretischen Informatik sowie der Logik. Aber trotz seiner engen Verwandtschaft zur Unifikation ist Semi-Unifikation ein deutlich schwierigeres Problem, wie Kfoury, Tiuryn und Urzyczyn Anfang der neunziger Jahre durch ihren Unentscheidbarkeitsbeweis für Semi-Unifikation zeigten. Ein wichtiges Werkzeug in diesem Beweis sind so genannte *Pfadgleichungen* (path equations), welche eine algebraische Darstellung von sowohl Turing Maschinen als auch Semi-Unifikationsproblemen erlauben.

Diese Arbeit präsentiert eine modifizierte Version des originalen Unentscheidbarkeitsbeweises, welche die Rolle und Ausdrucksstärke von Pfadgleichungen ausführlicher beleuchtet als das Original. Es wird gezeigt, dass nicht nur jedes Semi-Unifikationsproblem in ein System von Pfadgleichungen übersetzt werden kann, sondern umgekehrt auch jedes System von Pfadgleichungen in ein äquivalentes Semi-Unifikationsproblem. Das beweist, dass obwohl Pfadgleichungen oft eine bequemere Repräsentation sind, ihre Ausdrucksstärke jene von Semi-Unifikationsproblemen nicht übersteigt. Des Weiteren wird gezeigt, dass Systeme von Pfadgleichungen, so wie Semi-Unifikations- und Unifikationsprobleme auch, sofern sie überhaupt lösbar sind, immer eine *allgemeinste* Lösung besitzen.

Abschließend werden Pfadgleichungen auf erweiterte Semi-Unifikationsprobleme angewandt, in denen die lösenden Substitutionen auch Terme *unendlicher* Tiefe einführen dürfen (diese werden dann unendliche Bäume genannt), und dadurch zwei bisher nicht publizierte Gegenbeispiele konstruiert. Das erste ist eine Instanz des Semi-Unifikationsproblems dessen Lösungen *zwingend* unendlich viele nicht-idente Subterme enthalten müssen (die Existenz solcher Instanzen war bereits bekannt, aber kein konkretes Beispiel). Das andere ist eine Instanz welche zwar unendliche Lösungen mit nur endlich vielen nicht-identen Subtermen besitzt, aber keine *allgemeinste* solche Lösung. Damit ist gezeigt, dass Semi-Unifikationsprobleme auf *diesem* Lösungsraum *nicht* immer allgemeinste Lösungen besitzen.

On the Expressibility of Semi-Unification Problems

Decidability and Undecidability Results

Master's Thesis

written by

Florian Pflug

supervised by

Ao.Univ.Prof. Dr.phil. Matthias Baaz

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Abstract

The *semi-unification problem* is a natural generalization of the well-known *unification problem*, and has applications in multiple fields of theoretical computer science as well as logic. But despite its close relationship to unification, semi-unification turned out to be a much harder problem than unification, and the general case was shown to be undecidable in the early nineties by Kfoury, Tiuryn and Urzyczyn. An important tool in that proof are *systems of path equations*, which can be viewed as algebraic representations of both certain kinds of Turing machines and semi-unification problems.

This thesis restates the original undecidability proof in a way that elucidates the role and expressive power of path equations more fully than the original proof does. In particular, it is shown that not only can every semi-unification problem be translated into a system of path equations, but that every system of path equations can, in turn, be translated back into an equivalent semi-unification problem. This shows that, while often being a more convenient representation, systems of path equations are not any more *expressive* than semi-unification already is. Systems of path equations are also shown to have the same *principality* property that both unification and semi-unification have, i.e. that the existence of any solution mandates the existence of a *most general* one.

Finally, path equations are applied to a generalized form of the semi-unification problems, in which the solving substitutions are allowed to introduce terms of *infinite* depth, called *infinite trees*. There, path equations are used to construct two novel counter-examples. One is an instance of the semi-unification problem whose solutions must *necessarily* have infinitely many non-identical subterms (the *existence* of such instances was previously established, but no particular example was known). The other is an instance which *does* have infinite solutions containing only finitely many non-identical subterms, but no *principal* such solution. This proves that the principality property *fails* on this solution space.

Contents

Chapter 1

Introduction

1.1 The History and Significance of Semi-Unification

Semi-unification is, at first sight, a simple generalization of the well-known unification algorithm. Unification answers the question, given a list of term pairs, of whether a single substitution exists which makes each "left" term equal to the corresponding "right" term. Semi-unification generalizes this by instead looking for a single, global, substitution which only makes each "right" term a *substitution instance* of the corresponding "left" term. Note that this introduces two kinds of substitutions – one global substitution which is applied to *every* term in the list, and one local substitution per term pair which is applied only to the "left" term. (See section 2.1 for a formal definition)

The question of semi-unifiability arose independently in multiple different fields, at roughly the same time. Baaz and Pudlák used it in [BP93] to prove a version of Kreisel's conjecture for a particular deduction system called $L\exists_1$, which states (in its general form) that whenever a deduction system T proves $A(S^n(0))$ uniformly in k steps, then T proves $\forall x\, A(x)$. Henglein showed in [Hen88] that typability in the Milner-Mycroft calculus – which allows general mutually recursive definitions of polymorphic functions – can be reduced to semi-unification. And semi-unification also plays a role in other fields, for example computational linguistics, see [KTU93] for a more complete list.

The informal definition above already shows that semi-unification is, as the name implies, closely related to unification. This relationship goes beyond the two definitions just being superficially similar. [Baa93] shows, for example, that just like unification problems, semi-unification problems always have a *most general* solution if they have a solution at all – and Baaz proves this by exploiting that semi-unification problems can be solved by *repeated* unification, where the variables occurring on the left-hand side are renamed between the individual unification steps.

This close relationship initially lead to the belief that semi-unification, just like unification is *decidable*, i.e. that there is some algorithm which, given a semi-unification problem, tells whether that problem has a solution or not. Algorithms which claimed to decide semi-unification where even published, e.g. [KTU88], but in each case turned out to either wrongly call some problems unsolvable which do in fact have a solution, or to not always terminate. [KTU93] then showed that these problems are not simply mistakes in the published algorithms, but that semi-unification is, contrary to what intuition might say, actually undecidable.

Quite surprisingly, the proof of semi-unification's undecidability does not exploit its connection to typability problems. Instead, it reduces questions about the termination properties of certain Turing machines, previously shown to be undecidable in [Hoo66], to semi-unification. The following thesis is an attempt at making this rather long-winded proof more accessible, and to emphasize the role that so-called *path equations* (see section 2.2) play in semi-unification's undecidability.

A natural generalization of semi-unification drops the requirement that solutions must only produce *finite* terms, and instead allows arbitrary *infinite* trees to be introduced. [DR92] shows that the resulting decision problem, i.e. whether a given semi-unification problem has such a generalized solution, to be undecidable in general as well. This undecidability result holds both for semi-unification over infinite trees with only *finitely* many non-identical subtrees, called *rational trees*, as well as for the general case of arbitrary infinite trees.

1.2 Overview of the Proof and Deviations from the published Proof

The proof of the semi-unification problem's (SUP) undecidability presented here is based on the proof by Kfoury at al., found in [KTU93]. It follows largely the same path, but with a few smaller and one relatively large deviation.

From Turing Machines to Semi-Unification

The crucial step in both proofs is a reduction of the *boundedness*[1] problem for a certain class of Turing machines called *symmetric inter-cell*[2] *Turing machines* (SITMs) to a semi-unification problem, such that the semi-unification problem has a solution if and only if the original machine is bounded. The main tool for proving this equivalency are systems of *path equations* (cf. definition 2.2.6) with corresponding rules of inference, introduced by [KTU93] and used herein in a slightly modified form.

In the original proof, SITMs are directly translated into semi-unification problems, and path equations plus their inference rules are then used to show that boundedness of the former is equivalent to solvability of the latter (cf. [KTU93] lemma 11). The proof presented here instead places path equations at the center of things, and uses a modified set of inference rules. Whereas in [KTU93] derivable path equations may "speak" about non-existing subterms, the deduction system used here (cf. definition 2.3.1) contains an additional restriction which prevents this. That allows a rather nice characterization of exactly the solvable systems of path equations via the existence of *bounds* on the *length* of paths that appear in derivable path equations (cf. definition 2.3.3 and theorem 2.4.15).

While lemma 4 of [KTU93] is somewhat similar to this characterization, it also highlights the main difference between the two approaches. Part 2 of that lemma establishes a connection between the substitutions carried out by a *semi-decision* algorithm for semi-unification, and is ultimately used to show that unbounded SITMs yield, after translation, unsolvable instances of semi-unification. Theorem 2.4.15 in this thesis has a similar role, but doesn't refer to any semi-decision algorithm. Instead, it constructs solution of what are called *bounded* systems of path equations in a more algebraic way, and applies to *arbitrary* sets of path equations instead of only those which arise from semi-unification problems.

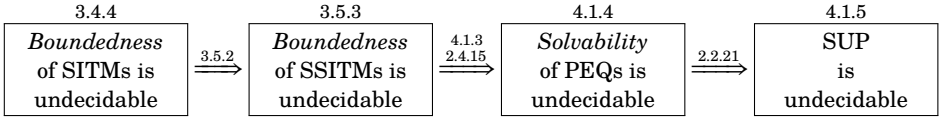
Following the idea of placing path equations at the center of things, this proof then first reduces the question of boundedness of a SITM not to a semi-unification problem, but rather to the solvability of some finite system of path equations (PEQs). For technical reasons, it does so in two steps. First, an SITM is brought into a form called *strict symmetric inter-cell Turing machine* (SSITM), which restricts the tape to contain a single (though arbitrarily long) non-blank region, surrounded by only blank symbols (cf. definition 3.5.1 and theorem 3.5.2). For this class of machines, the translation into a system of path equations is then almost trivial (cf. definition 4.1.1), and it is shown that the resulting system of path equations is bounded (i.e. solvable) exactly if the SSITM is bounded (cf. theorem 4.1.3).

Finally, to derive the final result about semi-unification, the solvability of finite sets of path equations is reduced to the solvability of semi-unification problems. This step, unfortunately, is about as onerous as the direct reduction from SITMs to instances of SUP in the original proof (cf. page 98 of [KTU93] and cf. theorem 2.2.21 of this thesis). At least, in the presented approach, one is rewarded for working through these irritations by a theorem that reduces *arbitrary* finite sets of path equations to instances of SUP, whereas the very similar reduction in the original proof is tailored only to the case of SITMs.

[1] *Boundedness* means that there is a global bound on the number of configurations that are reachable from an arbitrary initial configuration

[2] These are machines where the head is positioned *between* two tape cells, and whose reachability relation is *symmetric*

The following diagram shows the major steps of the part of the proof discussed so far – the reduction of the boundedness problem for SITMs to the semi-unification problem (SUP) – and show both the major theorems encountered along the way, as well as the theorems used in each of the implications.



From Mortality of TMs to Boundedness of SUP

The second part of both the proof presented here, and the one found in [KTU93], establishes the undecidability of boundedness for symmetric inter-cell Turing machines (the part is actually – in both works – found in the *middle*, but conceptually its still the second part that completes the proof).

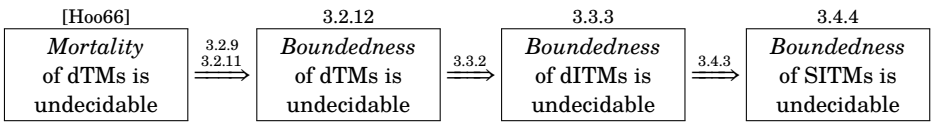
In this part, the difference between this text and [KTU93] lies mostly in the level of detail that is presented. [KTU93] is concerned more with brevity whereas this text contains explicit constructions of all the necessary reductions from one machine to another.

The proof starts from the undecidability of *mortality*[1] for deterministic Turing machines (dTMs), a result originally shown in [Hoo66]. By showing that unbounded machines are immortal and that mortality is decidable for bounded machines, it is then shown that *boundedness* of deterministic Turing machines is undecidable as well.

Unfortunately, the boundedness problem of deterministic Turing machines cannot easily be translated *directly* into semi-unification problems or systems of path equations – the most obvious obstacle being that such a translation requires a machine with a symmetric reachability relation. Constructing a more suitable class of machines – in a way that ensures that boundedness remains undecidable – is the goal of sections 3.3 and 3.4.

The main technical problem in that construction is that regular Turing machines cannot easily be "symmetricized" by simply replacing their successor relation $\vdash$ with its symmetric closure – if one does that, the resulting machine has little in common with the original one (cf. section 3.3 for a more detailed discussion). This proof uses the same, rather ingenious, "trick" as [KTU93] to overcome this issue. Regular Turing machines are first transformed into *inter-cell* Turing machines (ITMs)[2]. Since such machines can always read (and overwrite) whatever they wrote in the previous step (provided they move in the opposite direction), such machines *can* easily be "symmetricized", and this yields the already mentioned class of *symmetric inter-cell Turing machines* (SITMs). Interestingly, while every inter-cell Turing machine, deterministic or not, can be "symmetricized", only for *deterministic* machines is the resulting symmetric machine bounded exactly iff the original machine was bounded. This is the reason why it is necessary to establish the undecidability of boundedness specifically for *deterministic* machines, since only those can be transformed into an equivalent SITM and then further into an equivalent system of path equations or an equivalent instance of SUP.

The following diagram summarizes the main step in the argument that establishes the undecidability of boundedness for SITMs.



[1] A *mortal* machine always terminates, regardless of the initial configuration, even if the tape initially contains *infinitely* many non-blank cells

[2] Machines, where the head is positioned *between* two tape cells, and has the ability to read (and overwrite) *either* of them

Chapter 2

Semi-Unification and Path Equations

2.1 Basic Definitions

In the following, $\mathcal{V}$ denotes an infinite set of *variables*, "$\dot{\rightarrow}$" some binary function symbol and $\mathcal{E}$ the set of finite terms built from $\mathcal{V}$ and $\dot{\rightarrow}$. The following definition gives a more explicit construction of $\mathcal{E}$ which is necessary to formalize proofs about the elements of $\mathcal{E}$.

Definition 2.1.1 (Finite terms)**.**

(i) The set $\mathcal{E}$ of all finite terms *over $\mathcal{V}$ and $\dot{\rightarrow}$ is*

$$\mathcal{E} := \bigcup_{i \in \mathbb{N}} \mathcal{E}_i,$$

where

$$\mathcal{E}_0 := \mathcal{V},$$
$$\mathcal{E}_{i+1} := \mathcal{E}_i \cup \{\, (u \dot{\rightarrow} v) \mid u, v \in \mathcal{E}_i \,\}.$$

(ii) The maximal number of nested applications of $\dot{\rightarrow}$, or depth*, of a certain term $t \in \mathcal{E}$ is*

$$\operatorname{depth} e := \min \{\, i \in \mathbb{N} \mid e \in \mathcal{E}_i \,\}$$

Note that for every $t \in \mathcal{E}$, either $\operatorname{depth} e = 0$ *or there are e_l, e_r with $e = (e_l \dot{\rightarrow} e_r)$ and thus* $\operatorname{depth} e_l < \operatorname{depth} e$, $\operatorname{depth} e_r < \operatorname{depth} e$.

The function symbol $\dot{\rightarrow}$ is treated as right-associative, and extraneous brackets are skipped accordingly when writing out elements of $\mathcal{E}$. Thus, both "$\alpha \dot{\rightarrow} \beta \dot{\rightarrow} \gamma$" and "$\alpha \dot{\rightarrow} (\beta \dot{\rightarrow} \gamma)$" denote the same element of $\mathcal{E}$.

In the following, the focus lies on properties of terms and relationships between terms which are *independent* from any particular interpretation. A natural class of operators on the elements of $\mathcal{E}$ in this settings are substitutions – functions on terms which replace all occurrences of a particular variable $\alpha \in \mathcal{V}$ with some more specific term.

Definition 2.1.2 (Substitution)**.** *A function $S : \mathcal{E} \to \mathcal{E}$ is called a* substitution *on $\mathcal{E}$ if*

$$S(e_l \dot{\rightarrow} e_r) = S(e_1) \dot{\rightarrow} S(e_2) \quad \text{for all } e_l, e_r \in \mathcal{E}$$

The application of a substitution S to a term t is sometimes denoted by tS instead of $S(t)$. In the case of multiple substitutions, $tS_1 \ldots S_n$ stands for $S_n(\ldots S_1(t)\ldots)$. Note that the application order is *left to right* here, contrary to how the function concatenation operator $\circ$ is usually defined! In other words,

$$S_n(\ldots S_1(t)\ldots) = tS_1 \ldots S_n \overset{!}{\neq} t(S_1 \circ \cdots \circ S_n) = S_1(\ldots S_n(t)\ldots).$$

To minimize the risk of confusion, the syntax $t(S_1 \circ S_2)$ will not actually be used. The operator $\circ$ will only be used if the resulting substitution is not immediately applied to a term, as in $S' = \widetilde{S} \circ S$, and even that syntax will be used very sparingly.

Comparing the condition above to the construction of the set $\mathcal{E}$ shows that substitutions are uniquely defined by their actions on the set of variables $\mathcal{V}$.

Lemma 2.1.3. *For every $\mathring{S} : \mathring{\mathcal{V}} \to \mathcal{E}$ with $\mathring{\mathcal{V}} \subset \mathcal{V}$ there is a unique substitution S on $\mathcal{E}$ with $S|_{\mathring{\mathcal{V}}} = \mathring{S}$ and $S|_{\mathcal{V} \setminus \mathring{\mathcal{V}}} = \mathrm{id}$. S is called the* extension *of $\mathring{S}$ to $\mathcal{E}$.*

Proof. Define a series of partial functions $S_i : \mathcal{E}_i \to \mathcal{E}$ by setting $S_0 := \mathring{S} \cup \mathrm{id}_{\mathcal{V} \setminus \mathring{\mathcal{V}}}$ and

$$\begin{aligned} S_{i+1}(e) &:= S_i(e) && \text{if depth } e = i \\ S_{i+1}(e_l \dashrightarrow e_r) &:= (S_i(e_l) \dashrightarrow S_i(e_r)) && \text{if depth } e_l \dashrightarrow e_r = i+1. \end{aligned}$$

The definition

$$S(e) := S_{\text{depth } e}(e)$$

then yields a total function $\mathcal{E} \to \mathcal{E}$ which is a substitution. Since for all $v \in \mathring{\mathcal{V}}$ one has $S(v) = S_0(v) = \mathring{S}(v)$ it follows that $S|_{\mathcal{V}} = \mathring{S}$. For $v \in \mathcal{V} \setminus \mathring{\mathcal{V}}$, $S_0(v) = v$ by construction. ∎

Functions $\mathring{S} : \mathring{\mathcal{V}} \to \mathcal{E}$ with $\mathring{\mathcal{V}} \subset \mathcal{V}$ will often be called *substitutions* themselves, with the understanding that $\mathring{S}(t)$ for $t \in \mathcal{E}$ is to be read as $S(t)$ where S is the extension of $\mathring{S}$ to $\mathcal{E}$. In particular, if a function that is defined only on a proper subset of $\mathcal{V}$ is called a substitution, it is assumed to map all variables outside its defined range to themselves.

Take now a pair of terms $t, u \in \mathcal{E}$, and observe that for some such pairs of terms there exists a substitution which turns both into the same term, while for other pairs this is impossible. The pair $(\alpha, \beta \dashrightarrow \gamma)$ is an example of the first case, as the substitution $S(\alpha) = (\beta \dashrightarrow \gamma)$ proves. An example of the second case is the pair $(\alpha, \beta \dashrightarrow \alpha)$ (note that depth $S(\alpha) <$ depth $S(\beta \dashrightarrow \alpha)$, and hence $S(\alpha) \neq S(\beta \dashrightarrow \alpha)$, holds for every substitution S on $\mathcal{E}$). This motivates

Definition 2.1.4 (Term Equations)**.** *For two terms $t, u \in \mathcal{E}$ the expression*

$$t \doteq u$$

is called a term equation *or* unification *problem. A* solution *of such a term equation is a substitution S on $\mathcal{E}$ with $S(t) = S(u)$. If such a solution exists, it is also called a* unifier *of t and u.*

A natural generalization is to consider not only single term equations, but finite sets of such equations, and to look for substitutions which solve *all* equations in such a set *simultaneously*. While sounding harder initially, it turns out that this generalized case can be reduced to the original case of only one equation.

Theorem 2.1.5. *Let E be a finite set of term equations $\{t_1 \doteq u_1, \ldots, t_n \doteq u_n\}$. Then there exists a substitution S with $S(t_i) = S(u_i)$, $1 \leq i \leq n$ iff there exists a solution of the term equation*

$$t_1 \dashrightarrow t_2 \dashrightarrow \ldots \dashrightarrow t_{n-1} \dashrightarrow t_n \doteq u_1 \dashrightarrow u_2 \dashrightarrow \ldots \dashrightarrow u_{n-1} \dashrightarrow u_n$$

Proof.

$$\begin{array}{rrcl} & S(t_1 \dashrightarrow \ldots \dashrightarrow t_n) & = & S(u_1 \dashrightarrow \ldots \dashrightarrow u_n) \\ = & S(t_1) \dashrightarrow \ldots \dashrightarrow S(t_n) & = & S(u_1) \dashrightarrow \ldots \dashrightarrow S(u_n) \\ \Leftrightarrow & S(t_1) = S(u_1) & \ldots & S(t_n) = S(u_n) \end{array}$$

This equivalences prove both directions ∎

Again taking a pair of terms $t, u \in \mathcal{E}$, one can also ask whether u is the result of applying some substitution to t. This relation, contrary to the case of unifiability, is transitive in addition to being reflexive and therefore imposes a pre-order on the set $\mathcal{E}$.

Definition 2.1.6 (Instantiation Pre-Order)**.** *The pre-order defined by*

$$t \leq u \Leftrightarrow \textit{There exists a substitution } S \textit{ with } S(t) = u$$

is called the instantiation pre-order *on $\mathcal{E}$. If $t \leq u$, u is called an* instance *of t.*

Having defined an order relation on the set $\mathcal{E}$, term equations can be generalized to *term inequalities*.

Definition 2.1.7 (Term Inequalities)**.** *For two terms $t, u \in \mathcal{E}$ the expression*

$$t \dot{\leq} u$$

is called term inequality. *A* solution *of such a term inequality is a substitution S on $\mathcal{E}$ with $S(t) \leq S(u)$. If such a solution exists, it is also called a* semi-unifier *of t and u.*

Again, one can generalize that definition to the case of multiple inequalities. This, then, what is commonly called the *semi-unification problem*.

Definition 2.1.8 (Semi-Unification Problem)**.** *For n term inequalities*

$$\mathcal{S} = \left\{ t_i \dot{\leq} u_i \;\middle|\; 1 \leq i \leq n \right\}$$

the problem of whether a global substitution *S exists such that $S(t_i) \leq S(u_i)$ for all $1 \leq i \leq n$ is called* semi-unification *problem for $\mathcal{S}$. If such a solution exists, it is called* semi-unifier*, and the substitutions $S_1, \ldots, S_n$ for which $S_i(S(t_i)) = S(u_i)$ are called the* local substitutions *corresponding to S.*

For term equalities, theorem 2.1.5 shows that there is no significant difference between the case of a single term equation and the generalization to sets (systems) of such equations. That proof, however, does not carry over to the case of term inequalities, since it depends on the fact that $t_1 = u_1, t_2 = u_2$ holds exactly if $t_1 \dot{\rightarrow} t_2 = u_1 \dot{\rightarrow} u_2$ does. In the case of term inequalities, however, $t_1 \dot{\rightarrow} t_2 \leq u_1 \dot{\rightarrow} u_2$ is a stronger statement than $t_1 \leq u_1$, $t_2 \leq u_2$ since the former requires one substitution to make both pairs of terms equal while the latter allows for two different substitutions. This will prove to be not only a technical difficulty – it will be shown that the existence of a semi-unifier is undecidable in the general case of multiple inequalities, but decidable in the case of only one inequality.

2.2 Path Equations

In the following, it will be necessary to deal with partial functions defined on the set of terms $\mathcal{E}$. For this, it is convenient to extend $\mathcal{E}$ with a special undefined term, as carried out by

Definition 2.2.1 (Undefined Term)**.**

(i) The symbol $\perp$ denotes a special undefined term*, with $\perp \neq t$ for all $t \in \mathcal{E}$, and*

$$\mathcal{E}^{\perp} := \mathcal{E} \cup \{\perp\}.$$

(ii) A function $S : \mathcal{E}^{\perp} \to \mathcal{E}^{\perp}$ is called a substitution *if $S|_{\mathcal{E}}$ is a substitution and $S = S|_{\mathcal{E}} \cup \{\perp \to \perp\}$.*

Note that (ii) says that substitutions on $\mathcal{E}^{\perp}$ may neither map defined terms to undefined terms, nor undefined terms to defined ones! If a function defined on some *subset* of $\mathring{\mathcal{V}}$ is called a *substitutions* on $\mathcal{E}^{\perp}$, it is *not* assumed to map variables outside of $\mathring{\mathcal{V}}$ to $\perp$, but rather to themselves!

Theorem 2.1.5 showed that a set of term equations can be reduced to a single equivalent term equation. Now, in a way, the opposite is done. A given semi-unification problem is be transformed into a set of simpler equations which are equivalent to the original problem, in the sense that a substitution solves the resulting set of equations exactly if it solves the initial semi-unification problem.

The is the reason for introducing the function L and R, which "extract" the left respectively right subterm of a term.

Definition 2.2.2. *The functions $L,R : \mathcal{E}^{\perp} \to \mathcal{E}^{\perp}$ are defined as*

$$L : t \to \begin{cases} t_l & \text{if } t = t_l \dot{\to} t_r \\ \perp & \text{otherwise} \end{cases} \qquad R : t \to \begin{cases} t_r & \text{if } t = t_l \dot{\to} t_r \\ \perp & \text{otherwise} \end{cases}$$

By building strings from L and R, and also S_i, one can "speak" about the terms and their subterms that arise from the application of a *global* substitution S and particular *local* substitutions S_i.

Definition 2.2.3 (Path Expressions)**.**

(i) The sets of strings $\mathcal{P}$ (whose elements are called paths*) and $\mathcal{R}$ are*

$$\mathcal{P} := \{L, R\}^*, \quad \mathcal{R} := \{S_1, S_2, \ldots\}^*$$

(ii) The set of path expressions *$\mathcal{W}$ is*

$$\mathcal{W} := \{\, \Pi\mu\Sigma \mid \mu \in \mathcal{V},\ \Pi \in \mathcal{P},\ \Sigma \in \mathcal{R} \,\}$$

(iii) The interpretation *of a path expression*

$$\Pi\mu\Sigma = \pi_n \ldots \pi_1 \mu S_{i_1} \ldots S_{i_n} \in \mathcal{W}$$

under substitutions $S, S_1, S_2, \ldots$ is the term

$$\Pi((\mu S)\Sigma) := \pi_n\Big(\ldots \pi_1\Big(S_{i_n}\big(\ldots S_{i_1}(\mu)\ldots\big)\Big)\ldots\Big) \in \mathcal{E}^{\perp}.$$

Note that there is a slight ambiguity in the meaning of $\Pi\mu\Sigma$ – such an expression could be interpreted purely *syntactically* as a path expressions (i.e. an element of $\mathcal{W}$) or, for particular substitutions $S_1, \ldots$, as the *term* that results from applying Σ and then Π to the variable μ. This ambiguity is avoided by never omitting the brackets if an application of particular substitutions $S_1, \ldots$ is meant, and never including brackets in elements of $\mathcal{W}$.

This definition of *path expressions* does not allow expressions like $LS_1RS_2\mu$ where L,R and S_i application are interleaved. The following lemma shows that this does not restrict their generality, because all L and R applications can always be moved leftwards. The expression $LS_1RS_2\mu$, for example, then becomes $LR\mu S_1S_2$, and says exactly the same thing.

Lemma 2.2.4. *A function $S : \mathcal{E} \to \mathcal{E}$ is a* substitution *iff*

$$S(\Pi t) = \Pi S(t) \neq \perp \quad \text{for all } t \in \mathcal{E} \text{ and all } \Pi \in \mathcal{P} \text{ with } \Pi t \neq \perp.$$

Proof. Let t be an arbitrary compound term $t = t_l \dot{\rightarrow} t_r$. Then $Lt = t_l$, $Rt = t_r$ and thus

$$\begin{array}{rcl} S(t_l \dot{\rightarrow} t_r) & = & S(t_l) \dot{\rightarrow} S(t_r) \\ \Leftrightarrow \quad LS(t) = S(Lt) & \text{and} & RS(t) = S(Rt) \end{array}$$

This proves the lemma for $|\Pi| = 1$. The general case follows by induction on the length of Π. ■

The following clarifies how the usual concepts of *prefix pre-order* and empty strings apply to the sets $\mathcal{P}$ and $\mathcal{R}$. Note that elements of $\mathcal{P}$ are read right-to-left, while elements of $\mathcal{R}$ are read left-to-right, and the pre-orders are defined accordingly! This won't be a source of confusion (rather the opposite), because those strings usually appear in path expressions, where L and R are appended to the *left*, and the S_i are appended to the *right*.

Definition 2.2.5 (Prefix Order on $\mathcal{P}$ and $\mathcal{R}$)**.**

(i) The set $\mathcal{P}$ is partially ordered by

$$\Pi \leq \Pi' \quad \textit{if } \widetilde{\Pi}\Pi = \Pi' \textit{ for some } \widetilde{\Pi} \in \mathcal{P},$$

(ii) The set $\mathcal{R}$ is partially ordered by

$$\Sigma \leq \Sigma' \quad \textit{if } \Sigma\widetilde{\Sigma} = \Sigma' \textit{ for some } \widetilde{\Sigma} \in \mathcal{R}.$$

(iii) ε denotes the empty string and is an element of both $\mathcal{P}$ and $\mathcal{R}$.

(iv) If $p \leq e$ for some strings in $\mathcal{P}$ or $\mathcal{R}$, p is called a prefix *of e, and e an* extension *of p. In particular, ε is a prefix of every string, and every string extends ε.*

Path expressions allow "speaking" about the the terms and their subterms that arise from the application of a *global* substitution S and particular *local* substitutions S_i. What will mainly be "spoken *about*" in the following is which of these terms are equal.

Definition 2.2.6 (Path Equations)**.**

(i) A path equation *is an equation of the form*

$$\Pi_1\mu\Sigma_1 \doteq \Pi_2\nu\Sigma_2$$

where $\Pi_1\mu\Sigma_1$, $\Pi_2\nu\Sigma_2 \in \mathcal{W}$ are path expressions*.*

(ii) Substitutions $S, S_1, S_2, \ldots$ form a solution *of a set of path equations Γ if the interpretations of the path expressions on both sides of all the path equations agree, i.e. if*

$$\Pi_1\big((\mu S)\Sigma_1\big) = \Pi_2\big((\nu S)\Sigma_2\big) \neq \bot \quad \textit{for all } \Pi_1\mu\Sigma_1 \doteq \Pi_2\nu\Sigma_2 \in \Gamma.$$

Translating (Semi-)Unification Problems into Path Equations

For unification problems, path equations represent structural constraints on $S(t)$ and $S(u)$ which must be satisfied by some solution S to make it a solution of the unification problem. Thus, the variables occurring in these equations should be thought of as placeholders for the terms that some S might map them to, instead of variables in a purely syntactic sense.

Definition 2.2.7 (Path Equations for Unification Problems)**.** *For an arbitrary unification problem $t \doteq u$, the associated set of* path equations *is*

$$\begin{array}{rcl} \Gamma_{t \doteq u} & := & \left\{ \Pi\mu \doteq \nu \;\middle|\; \textit{There are paths } \Pi, \Pi' \textit{ with } \Pi' t = \mu \in \mathcal{V}, \Pi\Pi' u = \nu \in \mathcal{V} \right\} \\ & \cup & \left\{ \mu \doteq \Pi\nu \;\middle|\; \textit{There are paths } \Pi, \Pi' \textit{ with } \Pi\Pi' t = \mu \in \mathcal{V}, \Pi' u = \nu \in \mathcal{V} \right\}. \end{array}$$

Similarly, for semi-unification problems, path equations represent constraints on $S(t_i), S(u_i)$, $1 \leq i \leq n$, which must be satisfied by $S, S_1, \ldots, S_n$ to make them a solution of the semi-unification problem. Again, the variables are best thought of as placeholders for their images under S.

Definition 2.2.8 (Path Equations for Semi-Unification Problems)**.** *For an arbitrary semi-unification problem $\mathcal{S} := \left\{ t_i \preceq u_i \;\middle|\; 1 \leq i \leq n \right\}$, the associated set of* path equations *is*

$$\begin{array}{rcl} \Gamma_{\mathcal{S}} & := & \left\{ \Pi\mu S_i \doteq \nu \;\middle|\; \textit{There are paths } \Pi, \Pi' \textit{ with } \Pi' t_i = \mu \in \mathcal{V}, \Pi\Pi' u_i = \nu \in \mathcal{V} \right\} \\ & \cup & \left\{ \mu S_i \doteq \Pi\nu \;\middle|\; \textit{There are paths } \Pi, \Pi' \textit{ with } \Pi\Pi' t_i = \mu \in \mathcal{V}, \Pi' u_i = \nu \in \mathcal{V} \right\}. \end{array}$$

The following example demonstrates the relationship between a semi-unification problem and its associated set of path equations. Take the problem

$$(a \dashrightarrow a \dashrightarrow a) \dashrightarrow d \stackrel{\leq}{\cdot} d \dashrightarrow (b \dashrightarrow c),$$
$$b \stackrel{\leq}{\cdot} d.$$

The resulting set of path equations is then

$$aS_1 \stackrel{\cdot}{=} Ld, \quad aS_1 \stackrel{\cdot}{=} LRd, \quad aS_1 \stackrel{\cdot}{=} RRd,$$
$$LdS_1 \stackrel{\cdot}{=} b, \quad RdS_1 \stackrel{\cdot}{=} c,$$
$$bS_2 \stackrel{\cdot}{=} d.$$

It is obvious that the solution of a (semi-)unification problem also solves the associated set of path equations. The converse also holds, i.e. one has

Theorem 2.2.9. *If $t \stackrel{\cdot}{=} u$ is a unification problem with corresponding set of path equations $\Gamma_{t \stackrel{\cdot}{=} u}$, a substitution S solves $t \stackrel{\cdot}{=} u$ iff it solves $\Gamma_{t \stackrel{\cdot}{=} u}$.*

Theorem 2.2.10. *If $\mathcal{S} = \left\{ t_i \stackrel{\leq}{\cdot} u_i \;\middle|\; 1 \leq i \leq n \right\}$ is a semi-unification problem with corresponding set of path equations $\Gamma_{\mathcal{S}}$, substitutions S, $S_1, \ldots, S_n$ solve $\mathcal{S}$ iff they solve $\Gamma_{\mathcal{S}}$.*

The proof, however, is not immediately obvious. One needs to show that the set of path equations constrains a solution strongly enough to ensure that it also solves the original problem, which amounts to showing that it constrains "enough" paths. The first step is to formalize some facts about paths and their relationship to substitutions.

The first (rather trivial) such fact is that terms either have *no* subterm (if they are variables), or a left *and* a right subterm (if they are composite terms).

Lemma 2.2.11. *Let t be a term and Π be a path with $\Pi t \neq \bot$. Then for every Π' with $\Pi' < \Pi$ also $\Pi' t \neq \bot$, $L\Pi' t \neq \bot$ and $R\Pi' t \neq \bot$.*

Proof. If $\Pi' t$ was undefined, so would Πt be for every $\Pi \geq \Pi'$. For the rest, observe that if $\Pi' < \Pi$, then either $L\Pi' \leq \Pi$ or $R\Pi' \leq \Pi$, so at least one of them cannot be $\bot$. But then neither can the other be. ■

Consider now two terms $t, u \in \mathcal{E}$. Obviously, if $t = u$ then for every path Π one has $\Pi t = \Pi u$. The case $\Pi = \varepsilon$ shows that the converse is also obviously true. To prove 2.2.9 and 2.2.10, however, one needs this to hold more generally, since the set of path equations doesn't necessarily contain constraints for the path ε. The question is thus, can one find a proper subset W of $\mathcal{P}$, with $\varepsilon \notin W$, such that $\Pi t = \Pi u$ for all $\Pi \in W$ proves $t = u$? At first glance the case $t = \alpha$, $u = \beta$ for two distinct variables $\alpha, \beta \in \mathcal{V}$ is discouraging, since for these two terms $L(t) = L(u) = R(t) = R(u) = \bot$, yet $t \neq u$. This difficulty, however, can be overcome by taking the structure of t and u into account in the construction of W, as the following lemma shows.

Lemma 2.2.12. *Let $t, u \in \mathcal{E}$ be terms and $W \subset \mathcal{P}$ a set of paths where*

(i) for every $\Pi \in W$ at least either $\Pi t \neq \bot$ or $\Pi u \neq \bot$, and

(ii) for every $\Pi' \in \mathcal{P}$ there is a $\Pi \in W$ with either $\Pi \leq \Pi'$ or $\Pi' \leq \Pi$.

Then

$$t = u \neq \bot \quad \Leftrightarrow \quad \Pi t = \Pi u \neq \bot \quad \textit{for all } \Pi \in W$$

and W is called a witness for $t = u$.

Proof. If $t = u$ then obviously $\Pi t = \Pi u$ for all $\Pi \in \mathcal{P}$.

For the converse, assume that $t \neq u$. Then either $L(t) = R(t) = L(u) = R(u) = \bot$, or $L(t) \neq L(u)$ or $R(t) \neq R(u)$. Thus, by induction, one can find a path Π' such that $\Pi' t \neq \Pi' u$, but $\widetilde{\Pi} t = \widetilde{\Pi} u = \bot$ for all $\widetilde{\Pi} > \Pi'$.

By (ii) there is a $\Pi \in P$ with either $\Pi \leq \Pi'$ or $\Pi' \leq \Pi$. Since $\Pi t = \Pi u$ implies $\Pi' t = \Pi' u$ if $\Pi \leq \Pi'$, only the second case remains. Hence there is a $\Pi \in P$ with $\Pi' \leq \Pi$. But since Π' was constructed such that $\widetilde{\Pi} t = \widetilde{\Pi} u = \bot$ for all $\widetilde{\Pi} > \Pi'$, it follows from (i) that $\Pi' = \Pi$. Thus, there is a path $\Pi \in P$ with $\Pi t \neq \Pi u$. ■

Note that only condition (i) of lemma 2.2.12 takes the structure of t and u into account, and it really only requires t and u to be sufficiently "deep". Since applying a substitution never reduces a term's depth, the following lemma is unsurprising.

Lemma 2.2.13. *Let W be a witness for $t = u$ for $t, u \in \mathcal{E}$ and let S, T be an arbitrary substitution. Then W is also a witness for $S(t) = T(u)$.*

Proof. It suffices to show that condition (i) of lemma 2.2.12 still holds if t is replaced by $S(t)$. That in turn follows from the requirement that $S^{-1}(\bot) = \{\bot\}$ from the definition of substitutions on $\mathcal{E}^{\bot}$. ■

Lemma 2.2.12 did not provide a simply way to construct a suitable witness. This is now made up for by the following lemma, which shows that a suitable set is obtained by picking those paths which lead to variables.

Lemma 2.2.14. *Let $t, u \in \mathcal{E}$ be terms and let $W \subset \mathcal{P}$ be the set of paths Π with*

(i) $\Pi t \neq \bot$ *or* $\Pi u \neq \bot$, *and*

(ii) $\Pi t, \Pi u \in \mathcal{V} \cup \{\bot\}$.

Then W is a witness for $t = u$.

Proof. It it sufficient to show that condition (ii) of lemma 2.2.12 holds for W. Pick thus a $\Pi' \in \mathcal{P}$ which is not in W. Then, either $\Pi' t = \Pi' u = \bot$ or one of $\Pi' t, \Pi' u$ is from $\mathcal{E} \setminus \mathcal{V}$.

First case. There surely is a prefix $\Pi < \Pi'$ of Π' where either $\Pi t \neq \bot$ or $\Pi u \neq \bot$. For the longest such prefix Π, neither Πt nor Πu can be from $\mathcal{E} \setminus \mathcal{V}$, since otherwise the prefix could be extended by one. Thus, $\Pi \in W$ and by construction $\Pi < \Pi'$.

Second case. Pick the longest extension $\Pi > \Pi'$ of Π' where either $\Pi t \neq \bot$ or $\Pi u \neq \bot$. Then again neither Πt nor Πu can be in $\mathcal{E} \setminus \mathcal{V}$, since otherwise the longest extension could be extended by one while still maintaining $\Pi t \neq \bot$ or $\Pi u \neq \bot$. Thus, $\Pi \in W$ and by construction $\Pi' > \Pi$. ■

The similarly of the paths appearing in lemma 2.2.14 to the definitions 2.2.7 and 2.2.8 of the set of path equations corresponding to a (semi-)unification problem is of course no coincidence. It is precisely this similarity with ensures that a solution of one is also a solution of the other. The following proof exploits this.

Proof of theorem 2.2.10. For substitutions S and $S_1, \ldots, S_n$, to solve $\Gamma_{\mathcal{S}}$ means

$$\Pi_1\big((\mu S)S_i\big) = \Pi_2\big(\nu S\big) \neq \bot \qquad \text{for all equations } \Pi_1 \mu S_i \doteq \Pi_2 \nu \text{ in } \Gamma_{\mathcal{S}}.$$

which by definition 2.2.8 of $\Gamma_{\mathcal{S}}$ is the same as

$$\Pi(S_i(S(\mu))) = S(\nu) \neq \bot \qquad \text{for all paths } \Pi' \text{ with } \Pi' t_i = \mu,\ \Pi\Pi' u_i = \nu \text{ and}$$
$$S_i(S(\mu)) = \Pi S(\nu) \neq \bot \qquad \text{for all paths } \Pi' \text{ with } \Pi\Pi' t_i = \mu,\ \Pi' u_i = \nu,$$

and thus also as

$$\Pi(S_i(S(\Pi' t_i))) = S(\Pi\Pi' u_i) \neq \bot \qquad \text{for all paths } \Pi, \Pi' \text{ with } \Pi' t_i, \Pi\Pi' u_i \in \mathcal{V} \text{ and}$$
$$S_i(S(\Pi\Pi' t_i)) = \Pi S(\Pi' u_i) \neq \bot \qquad \text{for all paths } \Pi, \Pi' \text{ with } \Pi\Pi' t_i, \Pi' u_i \in \mathcal{V}.$$

Since $\Pi' t_i$ and $\Pi' u_i$ then cannot be undefined, their application commutes with the application of substitutions by lemma 2.2.4, and thus the previous holds if and only if

$$\Pi\Pi'(S_i(S(t_i))) = \Pi\Pi' S(u_i) \neq \bot \qquad \text{for all paths } \Pi, \Pi' \text{ with}$$
$$\Pi' t_i, \Pi\Pi' u_i \in \mathcal{V} \text{ or } \Pi\Pi' t_i, \Pi' u_i \in \mathcal{V}.$$

Setting $\widetilde{\Pi} = \Pi\Pi'$ and realizing that $\Pi' t_i, \Pi\Pi' u_i \in \mathcal{V}$ is equivalent to $\widetilde{\Pi} u_i \in \mathcal{V}$, $\widetilde{\Pi} t_i \in \mathcal{V} \cup \{\bot\}$ finally proves the equivalence of

$$\widetilde{\Pi}(S_i(S(t_i))) = \widetilde{\Pi} S(u_i) \neq \bot \qquad \text{for all paths } \widetilde{\Pi} \text{ with } \widetilde{\Pi} t_i, \widetilde{\Pi} u_i \in \mathcal{V} \cup \{\bot\}$$
$$\text{and either } \widetilde{\Pi} t_i \neq \bot \text{ or } \widetilde{\Pi} u_i \neq \bot.$$

Since these paths $\widetilde{\Pi}$ are exactly the paths in the witness W of lemma 2.2.14, the former holds if and only if

$$S_i(S(t_i)) = S(u_i) \qquad \text{for all } 1 \leq i \leq n$$

This series of equivalent rewriting proves both directions. ■

Proof of theorem 2.2.9. The proof of 2.2.10 works once all occurrences of the S_i are removed. ■

Translating Path Equations back into Semi-Unification Problems

Definition 2.2.8 and theorem 2.2.10 from the previous section together show that arbitrary semi-unification problems can be reduced to a solvability problem for some set of path equations. That raises the question of whether solvability of arbitrary sets of path equations is a more general (and more difficult) problem than semi-unification. For finite sets of path equations, it turns out that is it not – arbitrary finite sets of path equations can be translated (effectively!) back into some equivalent semi-unification problem.

Reducing arbitrary finite sets of path equations to semi-unification problems becomes technically a bit simpler if semi-unification is first generalized to include not only term inequalities, but also term equations. Additionally, instead of allowing different local substitutions to solve each inequality individually, the inequalities are now *tagged* with an index which specifies which local substitutions is supposed to solve it.

Definition 2.2.15 (Generalized Semi-Unification Problem)**.** *For finitely many equations and tagged term inequalities*

$$\mathcal{S} = \left\{ t_i \dot{\leq}_{l_i} u_i \;\middle|\; 1 \leq i < m \right\} \cup \left\{ t_i \doteq u_i \;\middle|\; m \leq i \leq n \right\}$$

with tags $l_i \in \{1, \ldots, k\}$*, the problem of whether a global substitution (or semi-unifier)* S *exists such that* $S(t_i) = S(u_i)$ *for all* $m \leq i \leq n$*, and and local substitutions* $S_1, \ldots, S_k$ *exists such that* $S_{l_i}(S(t_i)) = S(u_i)$ *for all* $1 \leq i < m$*, is called the* generalized semi-unification problem *for* $\mathcal{S}$*.*

Such generalized semi-unification problems can always easily (and effectively) be translated back into the form of definition 2.1.8, such that every solution of such a translation also solves the original, generalized problem. Since such a translation must introduce a new variable for every equations that appears in a generalized semi-unification problem, the reverse is, in general, not true – a solution of a generalized semi-unification problem will usually not solve the translated problem. Every solution of the generalized problem can, however, be easily *transformed* to include the newly introduced variables, and the transformed solution then *does* solve both the original, generalized problem as well as its translation to the form of definition 2.1.8.

Theorem 2.2.16. *Every generalized semi-unification problem* $\mathcal{S}$ *can be effectively reduced to a semi-unification problem* $\widetilde{\mathcal{S}}$ *as in definition 2.1.8 such that*

(i) $\widetilde{\mathcal{S}}$ *contains as many inequalities as* $\mathcal{S}$ *contains distinct tags (but at least one).*

(ii) Every solution of $\widetilde{\mathcal{S}}$ *is also a solution of* $\mathcal{S}$*.*

(iii) Every solution of $\mathcal{S}$ *can be effectively transformed into a solution of* $\widetilde{\mathcal{S}}$*.*

Proof. Let $\mathcal{S}$ be a generalized semi-unification problem

$$\mathcal{S} = \left\{ t_i \dot{\leq}_{l_i} u_i \;\middle|\; 1 \leq i < m \right\} \cup \left\{ t_i \doteq u_i \;\middle|\; m \leq i \leq n \right\}.$$

where $l_i \in\subset \{1, \ldots, k\}$ for $1 \leq i < m$. The equalities in $\mathcal{S}$ are transformed into the pair of terms $\hat{t}$ and $\hat{u}$ by picking distinct variables $\alpha_m, \ldots, \alpha_n$ that do not appear in $\mathcal{S}$ and setting

$$\begin{aligned} \hat{t} &:= (\alpha_m \rightharpoonup \alpha_m) \doteq (\alpha_{m+1} \rightharpoonup \alpha_{m+1}) \doteq \ldots \doteq (\alpha_n \rightharpoonup \alpha_n), \\ \hat{u} &:= (t_m \rightharpoonup u_m) \doteq (t_{m+1} \rightharpoonup u_{m+1}) \doteq \ldots \doteq (t_n \rightharpoonup u_n). \end{aligned}$$

Assume then (without loss of generality) that there are either no inequalities in $\mathcal{S}$ (in which case k is zero), or that there is at least one inequality for every tag in $\{1, \ldots, k\}$ (that can always be achieved by renaming the tags). All t_i respectively u_i whose inequalities carry the *same* tag are then combined to form the terms $\widetilde{t}_l$ respectively $\widetilde{u}_l$ for $l \in \{1, \ldots, k\}$, and to the terms for tag 1 the term $\hat{t}$ respectively $\hat{u}$ is appended. This yields

$$\begin{aligned} \widetilde{t}_1 &:= t_{I_1^1} \doteq \cdots \doteq t_{I_{N^1}^1} \doteq \hat{t}, & \widetilde{t}_l &:= t_{I_1^l} \doteq \cdots \doteq t_{I_{N^l}^l} \quad (\text{for } l > 1), \\ \widetilde{u}_1 &:= u_{I_1^1} \doteq \cdots \doteq u_{I_{N^1}^1} \doteq \hat{u}, & \widetilde{u}_l &:= u_{I_1^l} \doteq \cdots \doteq u_{I_{N^l}^l} \quad (\text{for } l > 1), \end{aligned}$$

where $\{I_1^l, \ldots, I_{N^l}^l\}$ are the indices of those inequalities tagged with l. The terms $\widetilde{t}_l$ and $\widetilde{u}_l$ are then combined to form the semi-unification problem

$$\widetilde{\mathcal{S}} := \left\{ \widetilde{t}_l \leq \widetilde{u}_l \;\middle|\; 1 \leq l \leq k \right\}.$$

(ii). If $S, S_1, \ldots, S_k$ is a solution of $\widetilde{\mathcal{S}}$, $S_l(S(\widetilde{t}_l)) = S(\widetilde{u}_l)$ for every $l \in \{1, \ldots, k\}$. By construction of the terms $\widetilde{t}_l$ and $\widetilde{u}_l$, that requires that $S_l(S(t_i)) = u_i$ for all $1 \leq i < m$, and that $S_1(\alpha_i) = S(t_i) = S(u_i)$. $S, S_1, \ldots, S_k$ is therefore then also a solution of $\mathcal{S}$.

(iii). Let $S, S_1, \ldots, S_k$ be a solution of $\mathcal{S}$. Since that implies that $S_{l_i}(S(t_i)) = S(u_i)$ for all inequalities $t_i \doteq_{l_i} u_i$ in $\mathcal{S}$, by construction of the terms $\widetilde{t}_l$ and $\widetilde{u}_l$, such a solution then also satisfies $S_l(S(\widetilde{t}_l)) = S(\widetilde{u}_l)$ for $2 \leq l \leq k$. Furthermore, it must also satisfy $S(t_i) = S(u_i)$ for $m \leq i \leq n$ and $S_1(S(t_i)) = S(u_i)$ for $i \in \{I^1_1, \ldots, I^1_{N^1}\}$. Now assume without loss of generality that the variables $\alpha_m, \ldots, \alpha_n$ do not appear in the image of any other variables under S (if they do, these occurrences of $\alpha_m, \ldots, \alpha_n$ can be replaced by other, fresh variables). Also assume that S does not replace $\alpha_m, \ldots, \alpha_n$ (since these variables do not appear in $\mathcal{S}$, any such replacement done by S is unnecessary, and can be dropped). Setting

$$S_1(\alpha_i) := S(t_i) \quad \Big(= S(u_i)\Big) \qquad \text{for } m \leq i \leq n$$

ensures that $S_1(S(\widetilde{t}_1)) = S(\widetilde{u}_1)$, and hence that $S, S_1, \ldots, S_k$ is a solution of $\widetilde{\mathcal{S}}$. ■

To reduce an arbitrary finite set of path equations Γ to a generalized semi-unification problem $\mathcal{S}$, one needs to find terms which reflect the structure imposed by Γ on the terms $S(\mu)$, where μ is a variable that appears in Γ and S the global substitution of a solution of Γ. If, for example, $\Gamma = \{La \doteq \ldots\}$, then $S(a)$ needs at least have a left subterm. That, of course, implies that $S(a)$ also needs to have a *right* subterm, even though the path expression Ra does not appear in Γ. One therefore cannot expect to always find terms which *only* have (defined) subterms at the positions indicated by the paths occurring in Γ. On the other hand, to be able to translate every solution of Γ to a solution of $\mathcal{S}$, $\mathcal{S}$ must not include unnecessarily large terms. The following definition of $\mathcal{P}^{\mu\Sigma}_\Gamma$ strikes the necessary balance. For every $\mu\Sigma$, it contains only paths Π for which $\Pi(\mu S\Sigma) \neq \bot$ for any solutions $S, S_1, \ldots, S_k$ of Γ, but enough paths to find terms for every $\mu\Sigma$ which *exactly* match the structure outlined by $\mathcal{P}^{\mu\Sigma}_\Gamma$.

$\mathcal{P}^{\mu\Sigma}_\Gamma$ is also defined to be non-empty (i.e. to at least include the empty path ε) for every $\mu\Sigma$ which *partially* matches some path expression in Γ. This is necessary because Γ may contain path expressions like $aS_1S_2 = \ldots$, and since semi-unification problems cannot directly represent the application of multiple local substitutions, an additional variable is required which represents aS_1.

Definition 2.2.17. *The paths concerning $\mu\Sigma$ within the path equations Γ are*

$$\begin{aligned}\mathcal{P}^{\mu\Sigma}_\Gamma := \{ L\Pi, R\Pi \mid{}& \widetilde{\Pi}\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \Gamma \text{ or } \Pi'\nu\Sigma' \doteq \widetilde{\Pi}\Pi\mu\Sigma \in \Gamma \text{ for } \widetilde{\Pi} \neq \varepsilon, \Pi'\nu\Sigma' \in \mathcal{W} \} \\ \cup\ \{ \varepsilon \mid{}& \widetilde{\Pi}\mu\Sigma\widetilde{\Sigma} \doteq \Pi'\nu\Sigma' \in \Gamma \text{ or } \Pi'\nu\Sigma' \doteq \widetilde{\Pi}\mu\Sigma\widetilde{\Sigma} \in \Gamma \text{ for } \Pi'\nu\Sigma' \in \mathcal{W}, \widetilde{\Sigma} \in \mathcal{R}, \widetilde{\Pi} \in \mathcal{P} \}\end{aligned}$$

As explained above, the important properties of the sets $\mathcal{P}^{\mu\Sigma}_\Gamma$ are these.

Corollary 2.2.18. *For a set of path equations Γ and the sets $\mathcal{P}^{\mu\Sigma}_\Gamma$,*

(i) For all paths $\Pi \in \mathcal{P}$, $L\Pi \in \mathcal{P}^{\mu\Sigma}_\Gamma$ iff $R\Pi \in \mathcal{P}^{\mu\Sigma}_\Gamma$,

(ii) if $\Pi \in \mathcal{P}^{\mu\Sigma}_\Gamma$ then $\Pi\big((\mu S)\Sigma\big) \neq \bot$ for all solutions $S, S_1, \ldots$ of Γ.

These sets $\mathcal{P}^{\mu\Sigma}_\Gamma$ now serve as "blueprints" for the construction of a term for every path expression in a set of path equations Γ.

Lemma 2.2.19. *For a finite set of path equations Γ, there exists an embedding $\theta_\Gamma : \mathcal{W} \to \mathcal{E}^\bot$ of the path expressions over variables appearing in Γ into the set of (possibly undefined) terms $\mathcal{E}^\bot$, which satisfies*

(i) $\theta_\Gamma\big(\Pi\mu\Sigma\big) \neq \bot$ exactly if $\Pi \in \mathcal{P}^{\mu\Sigma}_\Gamma$,

(ii) $\theta_\Gamma\big(\Pi\omega\big) = \Pi\theta_\Gamma\big(\omega\big)$ for all $\Pi \in \mathcal{P}$,

(iii) $\theta_\Gamma\big(\omega\big) = \theta_\Gamma\big(\omega'\big) \neq \bot$ only if $\omega = \omega'$,

(iv) All the terms produced by θ_Γ contain only variables that do not appear in Γ.

(v) $\theta_\Gamma(\Pi\mu\Sigma)$ can be determined effectively.

Proof. Assume that $\mathcal{V}$ contains, for every variable μ that appears in Γ, the (countably many) distinct variables μ^{Σ}_{Π} where $\Pi \in \mathcal{P}$ and $\Sigma \in \mathcal{R}$ (note that this implies that $\mu^{\varepsilon}_{\varepsilon}$ and μ are *distinct* variables!). Since Γ is finite, all the $\mathcal{P}^{\mu\Sigma}_{\Gamma}$ are finite, and the following recursive definition thus always hits a ground case in finitely many steps, making it well-defined and effective. It therefore indeed produces elements of $\mathcal{E}^{\perp}$, and these terms can easily be seen to satisfy (i)-(iv). The structure of $\mathcal{P}^{\mu\Sigma}_{\Gamma}$ guarantees that the cases are indeed exhaustive.

$$\theta_\Gamma(\Pi\mu\Sigma) := \begin{cases} \mu^{\Sigma}_{\Pi} & \text{if } \Pi \in \mathcal{P}^{\mu\Sigma}_{\Gamma},\ \widetilde{\Pi}\Pi \notin \mathcal{P}^{\mu\Sigma}_{\Gamma} \text{ for all } \widetilde{\Pi} \neq \varepsilon \\ \theta_\Gamma(L\Pi\mu\Sigma) \dashv\!\!\rightarrow \theta_\Gamma(R\Pi\mu\Sigma) & \text{if } \{L\Pi, R\Pi\} \subset \mathcal{P}^{\mu\Sigma}_{\Gamma}, \\ \perp & \text{otherwise.} \end{cases}$$

■

The terms $\theta_\Gamma(\Pi\mu\Sigma) \neq \perp$ are then used in the translations of arbitrary sets of path equations to semi-unification problems.

Theorem 2.2.20. *Every finite set of path equations Γ can be effectively reduced to a generalized semi-unification problem $\mathcal{S}_\Gamma$ such that*

(i) $\mathcal{S}_\Gamma$ contains as many distinct tags as Γ mentions distinct S_i,

(ii) every solution of $\mathcal{S}_\Gamma$ is also a solution of Γ,

(iii) every solution of Γ can be effectively transformed into a solution of $\mathcal{S}_\Gamma$.

Proof. With θ as in lemma 2.2.19, the translation of a finite Γ into a generalized semi-unification problem is carried out by

$$\begin{aligned} \mathcal{S}_\Gamma := \Big\{ \quad \theta_\Gamma(\mu\Sigma) \doteq_i \theta_\Gamma(\mu\Sigma S_i) \ \Big|\ \mathcal{P}^{\mu\Sigma S_i}_{\Gamma} \neq \emptyset \Big\} \qquad & ① \\ \cup \Big\{ \quad \mu \doteq \theta_\Gamma(\mu) \quad \Big|\ \mathcal{P}^{\mu}_{\Gamma} \neq \emptyset \Big\} \qquad & ② \\ \cup \Big\{ \theta_\Gamma(\Pi\mu\Sigma) \doteq \theta_\Gamma(\Pi'\nu\Sigma') \ \Big|\ \Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \Gamma \Big\} \qquad & ③ \end{aligned}$$

which, since only finitely many $\mathcal{P}^{\mu\Sigma}_{\Gamma}$ can be non-empty, is clearly an effective process. The role of ① is to enforce that the terms $\theta_\Gamma(\mu\Sigma S_i)$ are indeed the images of $\theta_\Gamma(\mu\Sigma)$ under S_i. ② merely establishes a connection between the auxiliary variables occurring in the terms produced by θ_Γ, and original variables from the set of path equations Γ. Last, but certainly not least, ③ includes all the constraints of Γ into $\mathcal{S}_\Gamma$, but expressed in terms of the auxiliary variables introduced by θ_Γ.

(i). Satisfied because $\mathcal{P}^{\mu\Sigma\S i}_{\Gamma} \neq \emptyset$ in particular requires that S_i is mentioned in Γ, cf. definition 2.2.17.

(ii). Let $S, S_1, \ldots, S_k$ be a solution of $\mathcal{S}_\Gamma$, and assume that $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \Gamma$. Then

$$\begin{array}{rcl} \perp \overset{\text{2.2.19 (i)}}{\neq} \theta_\Gamma(\Pi\mu\overbrace{S_{i_1}\ldots S_{i_k}}^{=\Sigma})S & \overset{③}{=} & \theta_\Gamma(\Pi'\nu\overbrace{S_{j_1}\ldots S_{j_{k'}}}^{=\Sigma'})S \\ \overset{\text{2.2.19 (ii)}}{=} \Pi\big(\theta_\Gamma(\mu S_{i_1}\ldots S_{i_k})S\big) & = & \Pi'\big(\theta_\Gamma(\nu S_{j_1}\ldots S_{j_{k'}})S\big) \\ \overset{①}{=} \Pi\big(\theta_\Gamma(\mu S_{i_1}\ldots S_{i_{k-1}})SS_{i_k}\big) & = & \Pi'\big(\theta_\Gamma(\nu S_{j_1}\ldots S_{j_{k'-1}})SS_{i_{k'}}\big) \\ \vdots & & \vdots \\ \overset{①}{=} \Pi\big(\theta_\Gamma(\mu)SS_{i_1}\ldots S_{i_k}\big) & = & \Pi'\big(\theta_\Gamma(\nu)SS_{j_1}\ldots S_{j_{k'}}\big) \\ \overset{②}{=} \Pi(\mu S\Sigma) & = & \Pi'(\nu S\Sigma') \end{array}$$

and $S, S_1, \ldots, S_k$ thus satisfies $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'$. Since that is true for every $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \Gamma$, $S, S_1, \ldots, S_k$ solves Γ.

(iii). Let the variables in the terms produced by θ_Γ be named μ^{Σ}_{Π} as in the proof of lemma 2.2.19, and let $S, S_1, \ldots, S_k$ be a solution of Γ. By construction, none of the variables μ^{Σ}_{Π} appear in Γ, but they could be *introduced* by one or more of the substitutions $S, S_1, \ldots, S_k$. If that is the case, all occurrences of such variables in $S, S_1, \ldots, S_k$ are replaced by fresh variables from $\mathcal{V}$ that do not clash with any of the variables in the terms θ_Γ. Note that $S, S_1, \ldots, S_k$ still form a solution of Γ after such a replacement.

Since $S, S_1, \ldots, S_k$ is a solution of Γ, by construction of the sets $\mathcal{P}_{\mu\Sigma}$, $\Pi(\mu S\Sigma) \neq \bot$ for all $\Pi \in \mathcal{P}_{\mu\Sigma}$. These substitutions may thus be extended by setting (remember that μ and μ^{Σ}_{Π} are *distinct* variables!)

$$S\left(\mu^{\Sigma}_{\Pi}\right) := \Pi(\mu S\Sigma).$$

for every μ that occurs in Γ. These extended substitutions then satisfy

$$S\big(\theta_\Gamma(\Pi\mu\Sigma)\big) = \Pi\big(\mu S\Sigma\big) \neq \bot$$

for every μ in Γ and every $\Pi \in \mathcal{P}_{\mu\Sigma}$. It remains to be shown that this suffices to make them fulfill the inequalities in $\mathcal{S}_\Gamma$ stemming from ①, and the equalities stemming from ② and ③.

①. Applying the extended S to an inequality from ① yields the inequality $\mu\Sigma \leq_i \mu\Sigma S_i$, which is clearly satisfied by S_i.

②. Applying the extended S to an equality from ② yields $\mu S = \mu S$ which is obviously true.

③. Applying the extended S to an equality from ③ yields $\Pi(\mu S\Sigma) = \Pi'(\nu S\Sigma')$, which must be true since $S, S_1, \ldots, S_k$ solves Γ. ■

This answers the question posed above of whether solvability of arbitrary finite sets of path equations is a more general problem than semi-unification.

Theorem 2.2.21. *The semi-unification problem for k inequalities is effectively equivalent to the solvability problem for finite sets of path equations mentioning k local substitutions.*

Proof. According to definition 2.2.8 every semi-unification problem $\mathcal{S}$ containing k inequalities can be transformed into a set of path equations $\Gamma_{\mathcal{S}}$ mentioning k local substitutions, and per 2.2.10 both problems have the same set of solutions.

Conversely, every finite set of path equations Γ mentioning k distinct local substitutions can, per theorem 2.2.20, be effectively reduced to a generalized semi-unification problem $\mathcal{S}_\Gamma$ with k distinct inequality tags. By theorem 2.2.16, that generalized semi-unification problem can then be reduced further to a plain semi-unification problem $\widetilde{\mathcal{S}}_\Gamma$ containing k inequalities. These theorems also show that every solution of $\widetilde{\mathcal{S}}_\Gamma$ is a solution of Γ, and that every solution of Γ can be effectively translated into a solution of $\widetilde{\mathcal{S}}_\Gamma$. ■

2.3 Boundedness of Path Equations

The *raison d'être* of path equations is that they provide insight into the existence of a solution of semi-unification problems. It does not suffice, however, to just look at the *original* path equations Γ corresponding to some semi-unification problem. Instead, one must include all those path equations $\overline{\Gamma}$ which are *derivable* from the original set. Since term equality is, like any sensible concept of equality, symmetric and transitive, $\overline{\Gamma}$ will at least have to include the symmetric and transitive closure of Γ. It will also have to be closed under applications of S_i to both sides of a path equation, i.e. for every path equation $\Pi_1\mu_1\Sigma_1 \doteq \Pi_2\mu_2\Sigma_2$ contain $\Pi_1\mu_1\Sigma_1 S_i \doteq \Pi_2\mu_2\Sigma_2 S_i$ because substitutions (as well-defined maps on the set of terms $\mathcal{E}$) map equal *defined* (i.e. $\neq \bot$) terms to equal *defined* terms.

But that alone doesn't still doesn't capture all the facts that can be inferred from Γ. Since equal *composite* terms have equal subterms, the path equation $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'$ also implies $L\Pi\mu\Sigma \doteq L\Pi'\nu\Sigma'$ and $R\Pi\mu\Sigma \doteq R\Pi'\nu\Sigma'$. Unless, that is, $\Pi((S(\mu))\Sigma)$ and $\Pi'((S(\nu))\Sigma')$ turn out to be variables, not composite terms. Thus, appending L or R to an existing path equation requires a proof that this is not the case. This is the role of the additional premise in the π-Application rule of the following definition of the deduction system for term equations.[1]

Definition 2.3.1. *(Derivable Path Equations) For a set Γ of path equations, $\overline{\Gamma}$ is the closure of Γ under the following deduction system, and is called the set of* derivable path equations*. ($\pi \in \{L, R\}$, $i \in \mathbb{N}$)*

$$\frac{\Pi_1\mu_1\Sigma_1 \doteq \Pi_2\mu_2\Sigma_2}{\Pi_2\mu_2\Sigma_2 \doteq \Pi_1\mu_1\Sigma_1} \qquad \text{(Symmetricity)}$$

$$\frac{\Pi_1\mu_1\Sigma_1 \doteq \Pi\nu\Sigma \qquad \Pi\nu\Sigma \doteq \Pi_2\mu_2\Sigma_2}{\Pi_1\mu_1\Sigma_1 \doteq \Pi_2\mu_2\Sigma_2} \qquad \text{(Transitivity)}$$

$$\frac{\Pi_1\mu_1\Sigma_1 \doteq \Pi_2\mu_2\Sigma_2}{\Pi_1\mu_1\Sigma_1 S_i \doteq \Pi_2\mu_2\Sigma_2 S_i}\, S_i \qquad (S_i\text{-Application})$$

$$\frac{\Pi_1\mu_1\Sigma_1 \doteq \Pi_2\mu_2\Sigma_2 \quad \left[\widetilde{\Pi}_2\pi\Pi_2\mu_2\Sigma_2 \doteq \Pi'\mu'\Sigma'\right]}{\pi\Pi_1\mu_1\Sigma_1 \doteq \pi\Pi_2\mu_2\Sigma_2}\, \pi \qquad (\pi\text{-Application})$$

In the following, multiple consecutive invocations of S_i-Application respectively π-Application will sometimes be contracted into one, and will be labelled Σ-Application respectively Π-Application for $\Sigma \in \mathcal{R}$ and $\Pi \in \mathcal{P}$. Similarly, a derivation of $\Pi_1\mu_1\Sigma_1 = \Pi_{n+1}\mu_{n+1}\Sigma_{n+1}$ from $\Pi_i\mu_i\Sigma_i = \Pi_{i+1}\mu_{i+1}\Sigma_{i+1}$ for $i \in \{1, \ldots, n\}$ by $n-1$ applications of Transitivity is sometimes denoted by a single deduction with the n premises $\Pi_i\mu_i\Sigma_i = \Pi_{i+1}\mu_{i+1}\Sigma_{i+1}$.

Lemma 2.3.2. *For an arbitrary set of path equations Γ, substitutions $S, S_1, \ldots$ solve Γ iff they solve $\overline{\Gamma}$.*

Proof. Since $\overline{\Gamma} \supset \Gamma$, it is clear that $S, S_1, \ldots$ solve Γ if they solve $\overline{\Gamma}$. For the converse, it suffices to check that all of the deduction rules are correct.

Symmetricity,Transitivity. Their correctness follows from the transitivity and symmetricity of term equality.

S_i-Application. The correctness follows from the well-definedness of substitutions as maps from $\mathcal{E}$ to $\mathcal{E}$, and the requirement that substitutions map *defined* terms to *defined* terms.

π-Application. Assume the assertion holds for the premise, meaning

$$\Pi_1(\Sigma_1(S(\mu_1))) = \Pi_2(\Sigma_2(S(\mu_2))) \neq \bot,$$

$$\widetilde{\Pi}_2\pi\Pi_2(\Sigma_2(S(\mu_2))) = \Pi'(\Sigma'(S(\mu))) \neq \bot.$$

From the latter it follows that $\pi\Pi_2(\Sigma_2(S(\mu_2))) \neq \bot$ and therefore also that

$$\pi\Pi_1(\Sigma_1(S(\mu_1))) = \pi\Pi_2(\Sigma_2(S(\mu_2))) \neq \bot.$$

For deductions containing more than one invocation of a rule, the result follows by induction. ■

[1] The deduction system used in the original proof (cf. [KTU93], page 93) is identical except that its *subterm* rule does not include such an additional premise.

Consider now the following unsolvable semi-unification problem and its corresponding set of path equations

$$\mathcal{S} := \{\alpha \rightharpoonup \beta \leqq \alpha\}, \qquad \Gamma_{\mathcal{S}} := \{\underbrace{\alpha S_1 \doteq L\alpha}_{=G_0}, \beta S_1 \doteq R\alpha\}.$$

The following proof scheme shows that all path equations G_n, $n \in \mathbb{N}$, of the form $LL^n\alpha \doteq L^n\alpha S_1$ are derivable from the path equation G_0, which is one of the two path equations corresponding to the original semi-unification problem.

$$\dfrac{\begin{array}{c} G_n \\ \downarrow \\ LL^n\alpha \doteq L^n\alpha S_1 \end{array} \qquad \dfrac{\begin{array}{c} G_n \\ \downarrow \\ LL^n\alpha \doteq L^n\alpha S_1 \end{array}}{[LL^n\alpha S_1 \doteq L^n\alpha S_1 S_1]}\, S_1}{\begin{array}{c} LL^{n+1}\alpha \doteq L^{n+1}\alpha S_1 \\ \downarrow \\ G_{n+1} \end{array}}\, L$$

Due to lemma 2.3.2, any solution of the original semi-unification problem would have to satisfy all these derived path equations, which in particular means the global substitution S of such a solution would have to satisfy $L^nS(\alpha) \neq \bot$ for *every* $n \in \mathbb{N}$. Since no finite term $S(\alpha)$ can accomplish that, it follows that Γ has no solution. Note, however, that some solvable systems of path equations also allow the derivation of path equations containing arbitrary long paths. For example, $\Gamma = \{L\alpha S_1 \doteq \alpha\}$ allows the derivation of $L^n\alpha S_1^n \doteq L^{n-1}\alpha S_1^{n-1}$ for arbitrary $n \in \mathbb{N}$. The difference is that the not all these paths apply to the same *term*

The following definition and lemma formalize this idea. Note that since $\overline{\Gamma}$ is closed under Symmetricity, the fact that it refers only to the left-hand sides of path equations is immaterial.

Definition 2.3.3 (Boundedness)**.** *For a set of path equations* Γ*,*

(i) The set Δ_Γ *containing all* derivable path expressions *is*

$$\Delta_\Gamma := \left\{ \Pi\mu\Sigma \;\middle|\; \Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma} \right\}$$

(ii) Γ *is called* bounded *if its set of derivable path expressions* Δ_Γ *contains, for every* $\mu \in \mathcal{V}$ *and* $\Sigma \in \mathcal{R}$*, only finitely many path expressions of the form* $\Pi\mu\Sigma$*, and* unbounded *otherwise.*

(iii) If Γ *is the set of path equations corresponding to a (semi-)unification problem, that problem is called* bounded *(respectively* unbounded*) if* Γ *is bounded (respectively unbounded).*

Lemma 2.3.4. *If* Γ *is a set of path equations and* $S, S_1, \ldots$ *one of its solutions,*

$$\Pi\mu\Sigma \in \Delta_\Gamma \quad \textit{implies} \quad \Pi\big((\mu S)\Sigma\big) \neq \bot$$

Proof. Follows from the fact that solutions of Γ are solutions of $\overline{\Gamma}$ (lemma 2.3.2) and the requirement that solutions may not fulfill path equations by mapping both sides to $\bot$ (definition 2.2.6). ■

From this, the one half of the indented equivalency of boundedness and solvability follows immediately

Theorem 2.3.5. *If a set of path equations* Γ *is unbounded, it has no solution.*

Proof. Let Γ be unbounded. Then there is a variable μ and a Σ such that Δ_Γ contains a sequence $\Pi_n\mu\Sigma$ with $|\Pi_n| \geq n$. Thus, if $S, S_1, \ldots$ were a solution of Γ, evaluating $(\mu S)\Sigma$ would need to yield a term with infinite depth due to lemma 2.3.4, since it would need to satisfy $\Pi_n(\mu S\Sigma) \neq \bot$ for all n. But that is impossible, hence Γ has no solution. ■

2.4 Solvability of Path Equations

The goal of this section is to explicitly construct a solution of a set of path equations, provided that this set is bounded. The crucial step in such a construction is, of course, to construct the *terms* that result from applying some substitution (either S, or one of the S_i) to a variable. Each derivable path expression (i.e., the elements of Δ_Γ) corresponds to such a term, or a subterm thereof. However, not every derivable path expression corresponds to a *distinct* subterm – derivable path *equations*, after all, say with subterms in a solution must necessarily be identical, so if $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'$ is derivable, the terms associated with $\Pi\mu\Sigma$ and $\Pi'\nu\Sigma'$ must be identical. A *natural* choice of set to construct the solution from is thus not Δ_Γ, but rather Δ_Γ *modulo* the path equations in $\overline{\Gamma}$, i.e. the derivable identities between path expressions. Unfortunately, however, that does not *quite* work out, for two reasons.

First, while each element of Δ_Γ *does* correspond to a term, or subterm, of the image of some variable under S or S_i, not *every* such subterm corresponds to an element of Δ_Γ. The set of path equations $\Gamma = \{La \doteq Lb, Ra \doteq Rb\}$, for example, has the solution $S(a) = S(b) = x \dot{\rightarrow} y$. Yet *no* element of $\Delta_\Gamma = \{La, Lb, Ra, Rb\}$ corresponds to this term – the elements all correspond to one of the *subterms* x and y. One could fix that by adding a new rule to the deduction system of path equations which allows $a \doteq b$ to be derived from $La \doteq Lb, Ra \doteq Rb$, but there is no real need for that. It is sufficient to instead use the set of *all* possible path equations $\mathcal{W}$, and partition *that* set modulo the derivable path equations. That will still yield distinct equivalence classes for a and b, but that doesn't matter. The two terms corresponding to those equivalence classes will automatically be identical, since $\overline{\Gamma}$ already constrains *both* their left and right subterms to be identical – in other words, La, Lb respectively Ra, Rb *will* lie in the same equivalence class.

Second, not *all* necessary identities do *directly* correspond to derivable path equations. Take for example the (very simple) system of path equations $\Gamma = \{a \doteq b, La \doteq c\}$. Note that Γ does not "speak" about Ra at all! But it *does* speak about La, meaning $L(aS)$ must be *defined* for a solutions S, and so any solution of Γ will still need to define $R(aS)$ as well – subterms can only have *no* subterm, or a left *and* a right subterm, after all. Such a solution is $S(a) = S(b) = c \dot{\rightarrow} d$. Note that this solution does not only *define* $R(aS)$, it also obeys $R(aS) = R(bS)$, even though $Ra \doteq Rb \neq \overline{\Gamma}$. And in fact, *every* solution of Γ will have to obey $R(aS) = R(bS)$, because otherwise $a = b$ wouldn't hold. Note that this problem is related to the first problem described above, but the *direction* is different – here, two "deeper" subterms need to be identical, but the path equations only constraint the terms "closer to the root", while above the subterms where provably identical and the identity of the upper-level terms followed automatically. That difference in direction is relevant, because terms are constructed *recursively* starting from their trivial subterms, which are variables. To avoid having to "look ahead" in the construction, the following definition folds that "look-ahead step" into the equivalence relation that is used to partition $\mathcal{W}$.

Definition 2.4.1 (Equivalent Path Expressions)**.** *The equivalence relation $\sim_\Gamma$ induced on $\mathcal{W}$ by a set of path equations Γ is*

$$\Pi\mu\Sigma \sim_\Gamma \Pi'\nu\Sigma' \quad \Leftrightarrow \quad \begin{cases} \Pi = \Pi',\ \mu = \nu,\ \Sigma = \Sigma' \text{ or} \\ \text{For some } \widetilde{\Pi} \in \mathcal{P},\ \Pi = \widetilde{\Pi}\hat{\Pi},\ \Pi' = \widetilde{\Pi}\hat{\Pi}' \text{ and } \hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma} \end{cases}$$

Proof that $\sim_\Gamma$ is indeed an equivalence relation. The relation $\sim_\Gamma$ is reflexiv by definition, and symmetric because $\overline{\Gamma}$ is (again by definition) closed under Symmetricity. That leaves the transitivity of $\sim_\Gamma$. Assume thus that $\Pi_1\mu_1\Sigma_1 \sim_\Gamma \Pi_2\mu_2\Sigma_2$ and $\Pi_2\mu_2\Sigma_2 \sim_\Gamma \Pi_3\mu_3\Sigma_3$. The situation is then as follows

$$\hat{\Pi}_1\mu_1\Sigma_1 \doteq \hat{\Pi}_2\mu_2\Sigma_2 \in \overline{\Gamma}, \qquad \Pi_1 = \widetilde{\Pi}\hat{\Pi}_1,\ \Pi_2 = \widetilde{\Pi}\hat{\Pi}_2 = \widetilde{\Pi}'\check{\Pi}_2,$$
$$\check{\Pi}_2\mu_2\Sigma_2 \doteq \check{\Pi}_3\mu_3\Sigma_3 \in \overline{\Gamma}, \qquad \Pi_3 = \widetilde{\Pi}'\check{\Pi}_3.$$

It follows from $\widetilde{\Pi}\hat{\Pi}_2 = \widetilde{\Pi}'\check{\Pi}_2$ that $\hat{\Pi}_2$ is either a prefix or an extension of $\check{\Pi}_2$, and it can be assumed without loss of generality that it's the latter, i.e. that $\hat{\Pi}_2 = \mathring{\Pi}\check{\Pi}_2$ for some $\mathring{\Pi}$. Substituting that into $\widetilde{\Pi}\hat{\Pi}_2 = \widetilde{\Pi}'\check{\Pi}_2$ yields $\widetilde{\Pi}\mathring{\Pi}\check{\Pi}_2 = \widetilde{\Pi}'\check{\Pi}_2$ and therefore $\widetilde{\Pi}\mathring{\Pi} = \widetilde{\Pi}'$. Thus, $\Pi_3 = \widetilde{\Pi}'\check{\Pi}_3 = \widetilde{\Pi}\mathring{\Pi}\check{\Pi}_3$, and the

derivation

$$\dfrac{\dfrac{\check{\Pi}_3\mu_3\Sigma_3 \doteq \check{\Pi}_2\mu_2\Sigma_2 \qquad [\overbrace{\mathring{\Pi}\check{\Pi}}^{\hat{\Pi}_2}\check{\Pi}_2\mu_2\Sigma_2 \doteq \hat{\Pi}_1\mu_1\Sigma_1]}{\mathring{\Pi}\check{\Pi}_2\mu_2\Sigma_2 \doteq \mathring{\Pi}\check{\Pi}_3\mu_3\Sigma_3}\,\mathring{\Pi} \qquad \overbrace{\mathring{\Pi}\check{\Pi}_2}^{\hat{\Pi}_2}\mu_2\Sigma_2 \doteq \hat{\Pi}_1\mu_1\Sigma_1}{\mathring{\Pi}\check{\Pi}_3\mu_3\Sigma_3 \doteq \hat{\Pi}_1\mu_1\Sigma_1}$$

shows $\overbrace{\widetilde{\Pi}\mathring{\Pi}\check{\Pi}_3}^{\Pi_3}\mu_3\Sigma_3 \sim_\Gamma \overbrace{\widetilde{\Pi}\hat{\Pi}_1}^{\Pi_1}\mu_3\Sigma_3$. ■

Having defined an equivalence relation on $\mathcal{W}$, the meaning of equivalence class (i.e. partition) follows naturally. Since equivalence classes contain path expressions, it will be convenient to use the same syntax path expressions use to append L, R to the left, and one of the S_i to the right. As always when one "pushes forward" a definition from a set to its set of cosets, one needs to check whether the resulting operation is well-defined.

Definition 2.4.2 (Equivalence Classes)**.** *For a set of path equations* Γ*,*

(i) The set $\mathcal{W}_\Gamma := \mathcal{W}/\sim_\Gamma$ *denotes the set of equivalent path expressions modulo* Γ*,*

(ii) The equivalence class $[\Pi\mu\Sigma]_\Gamma$ *of a path expression* $\Pi\mu\Sigma \in \mathcal{W}$ *is*

$$[\Pi\mu\Sigma]_\Gamma := \{\, \Pi'\nu\Sigma' \mid \Pi'\nu\Sigma' \sim_\Gamma \Pi\mu\Sigma \,\}.$$

(iii) For an equivalence class $[\Pi\mu\Sigma]_\Gamma \in \mathcal{W}_\Gamma$ *and arbitrary* $\widetilde{\Pi} \in \mathcal{P}, \widetilde{\Sigma} \in \mathcal{R}$*,*

$$\widetilde{\Pi}[\Pi\mu\Sigma]_\Gamma\widetilde{\Sigma} := [\widetilde{\Pi}\Pi\mu\Sigma\widetilde{\Sigma}]_\Gamma$$

Proof that $\widetilde{\Pi}[\Pi\mu\Sigma]_\Gamma\widetilde{\Sigma}$ *is well-defined by (iii).* If $\Pi\mu\Sigma \sim_\Gamma \Pi'\nu\Sigma'$, then $\hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma}$ such that $\Pi = \widetilde{\Pi}\hat{\Pi}$ and $\Pi' = \widetilde{\Pi}\hat{\Pi}'$ for some $\widetilde{\Pi} \in \mathcal{P}$. Applying S_i-Application to $\hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma'$ proves that $\Pi\mu\Sigma S_i \sim_\Gamma \Pi'\nu\Sigma' S_i$. That $L\Pi\mu\Sigma \sim_\Gamma L\Pi'\nu\Sigma'$ respectively $R\Pi\mu\Sigma \sim_\Gamma R\Pi'\nu\Sigma'$ is an immediate consequence of the definition of $\sim_\Gamma$. The general case follows by induction on the lengths of $\widetilde{\Pi}$ and $\widetilde{\Sigma}$. ■

As has been already said, the role of $\mathcal{W}_\Gamma$ will be to serve as a "template" along which the terms that appear in a solution of Γ will be constructed. To be able to do that, however, it needs to be established that $\mathcal{W}_\Gamma$ has a structure compatible with that of the set of terms $\mathcal{E}$. A defining property of terms is that they are non-circular – a terms can never be proper subterm of itself, or in the language of paths, $\Pi t = t$ only for $\Pi = \varepsilon$. The following lemma shows that the same holds for $\mathcal{W}_\Gamma$ – if Γ is bounded, that is. The failure of this lemma for some unbounded Γ (take e.g. $\Gamma = \{a \doteq La\}$) is, in fact, one of the reasons why unbounded sets of path equations (as was already shown) do *not* have solutions.

Lemma 2.4.3. *If a set of path equations* Γ *is bounded,* $\Pi\omega \neq \omega$ *for all* $\omega \in \mathcal{W}_\Gamma$ *and all* $\Pi \neq \varepsilon$*,*

Proof. If $\Pi\omega = \omega = [\check{\Pi}\mu\Sigma]_\Gamma \in \mathcal{W}_\Gamma$ then $\Pi\check{\Pi}\mu\Sigma \sim_\Gamma \check{\Pi}\mu\Sigma$, i.e. there is a $\hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\mu\Sigma \in \overline{\Gamma}$ such that for some $\widetilde{\Pi}$ one has $\Pi\check{\Pi} = \widetilde{\Pi}\hat{\Pi}$ and $\check{\Pi} = \widetilde{\Pi}\hat{\Pi}'$. Since $\Pi\check{\Pi} > \check{\Pi}$, it follows that $\hat{\Pi} > \hat{\Pi}'$, or in other words $\hat{\Pi} = \mathring{\Pi}\hat{\Pi}'$ for some $\mathring{\Pi} \neq \varepsilon$. For $n = 1$, the premises of the derivation

$$\dfrac{\overbrace{\mathring{\Pi}^n\hat{\Pi}'}^{\mathring{\Pi}^{n-1}\hat{\Pi}}\mu\Sigma \doteq \mathring{\Pi}^{n-1}\hat{\Pi}'\mu\Sigma \qquad [\overbrace{\mathring{\Pi}^n\hat{\Pi}'}^{\mathring{\Pi}^{n-1}\hat{\Pi}}\mu\Sigma \doteq \mathring{\Pi}^{n-1}\hat{\Pi}'\mu\Sigma]}{\mathring{\Pi}^{n+1}\mu\Sigma \doteq \mathring{\Pi}^n\hat{\Pi}'\mu\Sigma}\,\mathring{\Pi}$$

are simply $\hat{\Pi}\mu \doteq \hat{\Pi}'\mu$ and repeating it proves that $\mathring{\Pi}^n\hat{\Pi}'\mu\Sigma \in \Delta_\Gamma$ for arbitrary n. Since $\mathring{\Pi} \neq \varepsilon$, that contradicts Γ being bounded. ■

When using $\mathcal{W}_\Gamma$ as a template, i.e. when constructing terms "along" the structure of $\mathcal{W}_\Gamma$, one will need to know when to *stop*, i.e. when to insert a variable instead of a subterm. The example $\Gamma = \{La \doteq Ra, Lb \doteq Rb\}$ from above shows that doing so once one finds an equivalence class that does not contain any derivable path expression doesn't quite work – $[a]_\Gamma = \{a\}$ in this example, yet obviously a must be mapped to a composite term, not a variable. Yet this criterion is nearly

correct – the only required tweak is to look at *all* the equivalence classes "below", i.e. all those that can be produced by appending some path, and see if any of *those* are "spoken about" by Γ. Equivalence classes for which this isn't the case are, per the following definition, *terminal*.

Definition 2.4.4. *An equivalence class $\omega \in \mathcal{W}_\Gamma$ is called* terminal *if*

$$\widetilde{\Pi}\omega \cap \Delta_\Gamma = \emptyset \quad \textit{for all } \widetilde{\Pi} \neq \varepsilon,$$

and non-terminal *otherwise. $\dot{\mathcal{W}}_\Gamma \subset \mathcal{W}_\Gamma$ is the set of terminal equivalence classes.*

Terminality of an equivalence class ω is a property "inherited" by the images of ω under L and R.

Lemma 2.4.5.

(i) If $\omega \in \mathcal{W}_\Gamma$ is terminal, $\widetilde{\Pi}\omega$ is terminal for all $\widetilde{\Pi} \in \mathcal{P}$.

(ii) If $\Pi\omega \in \mathcal{W}_\Gamma$ is non-terminal, $\widetilde{\Pi}\omega$ is non-terminal for all $\widetilde{\Pi} \leq \Pi$.

Proof. Follows immediately from definition 2.4.4. ∎

The following lemma shows that images of terminal equivalence classes under L and R are unaffected by Γ – once the contents of some terminal ω is known, $\Pi\omega$ can be found without consulting Γ at all.

Lemma 2.4.6. *If $\omega \in \mathcal{W}_\Gamma$ is terminal, $\widetilde{\Pi}\omega = \{\, \widetilde{\Pi}\Pi\mu\Sigma \mid \Pi\mu\Sigma \in \omega \,\}$ for all $\widetilde{\Pi} \in \mathcal{P}$.*

Proof.

$\widetilde{\Pi}\omega \supset \{\, \widetilde{\Pi}\Pi\mu\Sigma \mid \Pi\mu\Sigma \in \omega \,\}$. Follows immediately from the definition 2.4.2 of $\widetilde{\Pi}\omega$.

$\widetilde{\Pi}\omega \subset \{\, \widetilde{\Pi}\Pi\mu\Sigma \mid \Pi\mu\Sigma \in \omega \,\}$. Assume that $\Pi' \nu\Sigma' \in \widetilde{\Pi}\omega$, or in other words that $\widetilde{\Pi}\Pi\mu\Sigma \sim_\Gamma \Pi'\nu\Sigma'$ for some $\Pi\mu\Sigma \in \omega$. It must be shown that $\Pi'\nu\Sigma' \in \{\, \widetilde{\Pi}\Pi\mu\Sigma \mid \Pi\mu\Sigma \in \omega \,\}$, i.e. that ⊛ there is a Π'' such that $\Pi' = \widetilde{\Pi}\Pi''$ and $\Pi''\nu\Sigma' \in \omega$.

Since $\widetilde{\Pi}\Pi\mu\Sigma \sim_\Gamma \Pi'\nu\Sigma'$, there is a $\hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma}$ with $\widetilde{\Pi}\Pi = \mathring{\Pi}\hat{\Pi}$ and $\Pi' = \mathring{\Pi}\hat{\Pi}'$ for some $\mathring{\Pi}$. The situation is thus that

$$\overbrace{\mathring{\Pi}\hat{\Pi}}^{\widetilde{\Pi}\Pi}\mu\Sigma \sim_\Gamma \overbrace{\mathring{\Pi}\hat{\Pi}'}^{\Pi'}\nu\Sigma' \quad \text{and} \quad \hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma}.$$

Note that $\widetilde{\Pi}\Pi = \mathring{\Pi}\hat{\Pi}$ implies that either $\hat{\Pi} > \Pi$ or $\hat{\Pi} \leq \Pi$.

If $\hat{\Pi} > \Pi$, i.e. $\hat{\Pi} = \Pi_1\Pi$ for some $\Pi_1 \neq \varepsilon$, then $\hat{\Pi}\mu\Sigma = \Pi_1\Pi\mu\Sigma \in \Pi_1\omega$. Since $\hat{\Pi}\mu\Sigma \in \Delta_\Gamma$ (due to $\hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma}$), that contradicts $\Pi_1\omega \cap \Delta_\Gamma = \emptyset$, i.e. that ω is terminal.

Therefore $\hat{\Pi} \leq \Pi$, i.e. $\Pi = \Pi_1\hat{\Pi}$ for some Π_1. Substituting into $\widetilde{\Pi}\Pi = \mathring{\Pi}\hat{\Pi}$ from above shows $\widetilde{\Pi}\Pi_1\hat{\Pi} = \mathring{\Pi}\hat{\Pi}$ and hence $\widetilde{\Pi}\Pi_1 = \mathring{\Pi}$. Applying that to $\Pi' = \mathring{\Pi}\hat{\Pi}'$ from above gives $\Pi' = \widetilde{\Pi}\Pi_1\hat{\Pi}'$, and setting $\Pi'' = \Pi_1\hat{\Pi}'$ yields the situation

$$\widetilde{\Pi}\underbrace{\overbrace{\Pi_1\hat{\Pi}}^{\Pi}\mu\Sigma}_{\in\omega} \sim_\Gamma \widetilde{\Pi}\underbrace{\overbrace{\Pi_1\hat{\Pi}'}^{\Pi'}}_{\Pi''}\nu\Sigma' \quad \text{and} \quad \hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma}.$$

Note that in particular, $\Pi'\nu\Sigma' = \widetilde{\Pi}\Pi''\nu\Sigma'$, thus fulfilling the first requirement put forth in ⊛. From $\omega \ni \Pi\mu\Sigma = \Pi_1\hat{\Pi}\mu\Sigma \sim_\Gamma \Pi_1\hat{\Pi}'\nu\Sigma' = \Pi''\nu\Sigma'$ (use Π_1 as the $\widetilde{\Pi}$ in the definition of $\sim_\Gamma$), it follows that $\Pi''\nu\Sigma' \in \omega$, which fulfills the second condition from ⊛ and thus concludes the proof. ∎

Having shown that terminal equivalence classes are "uninfluenced" by ω, now in a way the opposite is shown for non-terminal equivalence classes. Note that the following lemma is not a trivial consequence of the definition of terminality – the definition just requires a $\widetilde{\Pi} \neq \varepsilon$ such that *some* element of $\widetilde{\Pi}[\Pi\mu\Sigma]_\Gamma$ lies also in Δ_Γ, whereas the following shows that, in fact, $\widetilde{\Pi}\Pi\mu\Sigma$ *itself* lies in Δ_Γ. This will make reasoning about non-terminal equivalence classes a quite a bit simpler.

Lemma 2.4.7. *If $[\Pi\mu\Sigma]_\Gamma$ is non-terminal, $\widetilde{\Pi}\Pi\mu\Sigma \in \Delta_\Gamma$ for some $\widetilde{\Pi} \in \mathcal{P}$.*

Proof. If $\left[\Pi\mu\Sigma\right]_\Gamma$ is non-terminal, there is a $\widetilde{\Pi} \neq \varepsilon$ such that $\widetilde{\Pi}\Pi\mu\Sigma \sim_\Gamma \Pi' \nu\Sigma' \in \Delta_\Gamma$. It follows that there are $\mathring{\Pi}, \hat{\Pi}, \hat{\Pi}' \in \mathcal{P}$ which yield

$$\overbrace{\mathring{\Pi}\hat{\Pi}}^{\widetilde{\Pi}\Pi} \mu\Sigma \sim_\Gamma \underbrace{\overbrace{\mathring{\Pi}\hat{\Pi}\hat{\Pi}'}^{\Pi'} \nu\Sigma}_{\in\Delta_\Gamma} \quad \text{and} \quad \hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \in \overline{\Gamma}.$$

The derivation

$$\dfrac{\hat{\Pi}\mu\Sigma \doteq \hat{\Pi}'\nu\Sigma' \qquad \begin{array}{c}\mathring{\Pi}\hat{\Pi}\hat{\Pi}'\nu\Sigma \in \Delta_\Gamma \\ \downarrow \\ \left[\mathring{\Pi}\hat{\Pi}\hat{\Pi}'\nu\Sigma \doteq \ldots\right]\end{array}}{\underbrace{\mathring{\Pi}\hat{\Pi}}_{\widetilde{\Pi}}\mu\Sigma \doteq \mathring{\Pi}\hat{\Pi}'\nu\Sigma'}\,\mathring{\Pi}$$

shows that $\widetilde{\Pi}\mu\Sigma \in \Delta_\Gamma$ as required. ■

So far, the only statements pertaining the structure of $\mathcal{W}_\Gamma$ were lemma 2.4.3 and 2.4.5 – the other lemmata where concerned with the property of *single* equivalence classes and their relationship with Δ_Γ. But as was said already, to use $\mathcal{W}_\Gamma$ as a template, its structure must be shown to be sufficiently similar to that of $\mathcal{E}$, which will now be tackled.

The first such structural lemma concerns the relationship of $\left[\Pi\mu\right]_\Gamma$ and $\left[\Pi\mu S_i\right]_\Gamma$ (although it is stated in its generalized form). It confirms that one may indeed let non-terminal equivalence classes correspond to composite terms, because such equivalence classes yield other non-terminal equivalence classes when additional S_i are appended – just as substitutions map composite terms to composite terms.

Lemma 2.4.8. *If $\omega \in \mathcal{W}_\Gamma$ is non-terminal, $\omega\widetilde{\Sigma}$ is non-terminal for arbitrary $\widetilde{\Sigma} \in \mathcal{R}$.*

Proof. If $\omega = \left[\Pi\mu\Sigma\right]_\Gamma \in \mathcal{W}_\Gamma$ is non-terminal, then according to lemma 2.4.7 there is a $\widetilde{\Pi}$ such that $\widetilde{\Pi}\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma}$. Invoking $\widetilde{\Sigma}$-Application on this path equation yields $\widetilde{\Pi}\Pi\mu\Sigma\widetilde{\Sigma} \doteq \Pi'\nu\Sigma'\widetilde{\Sigma} \in \overline{\Gamma}$, thus in particular that $\widetilde{\Pi}\Pi\mu\Sigma\widetilde{\Sigma} \in \Delta_\Gamma$, and therefore that $\left[\Pi\mu\Sigma\widetilde{\Sigma}\right]_\Gamma = \omega\widetilde{\Sigma}$ is non-terminal. ■

It follows immediately from the definition of terminality that if $\left[\Pi\mu\Sigma\right]_\Gamma$ is non-terminal, all the equivalence classes "between" $\left[\mu\Sigma\right]_\Gamma$ and $\left[\Pi\mu\Sigma\right]_\Gamma$, i.e. $\left[\Pi'\mu\Sigma\right]_\Gamma$ with $\Pi' < \Pi$ are non-terminal. And all these equivalence classes, as will be seen, actually correspond to some subterm that appears in a solution of Γ. But there are, of course, very many *terminal* equivalence classes that do not play any relevant role in the construction of a solution – for $\Gamma = \left\{a \doteq a\right\}$, for example, $\left[La\right]_\Gamma$ is utterly irrelevant. Some terminal equivalence classes *are* relevant, though – it was already mentioned that exactly those equivalence classes will correspond to variables, and some of these variables must, of course, appears in a solution if it is to consist of well-formed terms. Formalizing the distinction between relevant and irrelevant equivalence classes is the gist of

Definition 2.4.9. *A $\Pi\omega \in \mathcal{W}_\Gamma$ is said to be* reachable *from $\omega \in \mathcal{W}_\Gamma$ if*

$$\widetilde{\Pi}\omega \text{ is non-terminal for all } \widetilde{\Pi} < \Pi,$$

and unreachable *from ω otherwise. Note that all non-terminal $\Pi\omega$ are reachable from ω, and that ω is always reachable from itself.*

In the interpretation of terminal equivalence classes as "variables" and non-terminal ones as "composite terms", the following lemma simply asserts that the "composite terms" all have finite depth (i.e. , are actually term as defined at the beginning).

Lemma 2.4.10. *For a bounded set of path equations Γ, there is a bound M_Γ^ω for every $\omega \in \mathcal{W}_\Gamma$ such that if $|\Pi| > M_\Gamma^\omega$ then $\Pi\omega$ is unreachable from ω.*

Proof. (ii). Assume $\omega = \left[\hat{\Pi}\mu\Sigma\right]_\Gamma$ and that $(\Pi_n)_{n\in\mathbb{N}}$ is a sequence of paths with $|\Pi_n| \geq n$ and where $\Pi_n\omega$ is reachable from ω for all n. Due to lemma 2.4.7, there would then exist a $\widetilde{\Pi}_n$ for every Π_n such that $\widetilde{\Pi}_n\Pi_n\hat{\Pi}\mu\Sigma \in \Delta_\Gamma$. Since $\left|\widetilde{\Pi}_n\Pi_n\hat{\Pi}\right| > |\Pi_n| \geq n$ that contradicts Γ being bounded. ■

In the proofs to come, some convenient way to prove two terms equal will be required. The basis for that was already introduced in lemma 2.4.11 with the concept of *witnesses*, and the following merely provides an easy way of finding such witnesses for the paths constructed from $\mathcal{W}_\Gamma$.

Lemma 2.4.11. *For a bounded set of path equations* Γ *and* $\omega \in \mathcal{W}_\Gamma$,

$$W_\omega := \left\{ \widetilde{\Pi} \in \mathcal{P} \;\middle|\; \widetilde{\Pi}\omega \text{ is terminal and reachable from } \omega \right\}$$

has the property (ii) of lemma 2.2.12, i.e. that

$$\text{for every } \Pi' \in \mathcal{P} \text{ there is a } \Pi \in W \text{ with either } \Pi \leq \Pi' \text{ or } \Pi' \leq \Pi.$$

Proof. Let $\Pi' \in \mathcal{P}$ be an arbitrary path.

If $\omega\Pi'$ is terminal, let $\mathring{\Pi}\tilde{\pi}\widetilde{\Pi} := \Pi'$ where $\tilde{\pi} \in \{L, R\}$ and $\widetilde{\Pi}$ is the *longest* prefix of Π' for which $\Pi'\omega$ is non-terminal. Then $\tilde{\pi}\widetilde{\Pi} \leq \Pi'$, and $\tilde{\pi}\widetilde{\Pi}\omega$ is terminal and reachable from ω, so $\tilde{\pi}\Pi\omega \in W_\omega$.

If $\omega\Pi'$ is non-terminal, lemma 2.4.10 implies that there are extensions $\mathring{\Pi} > \Pi'$ such that $\mathring{\Pi}\omega$ is unreachable from ω. Let $\tilde{\pi}\widetilde{\Pi} := \mathring{\Pi}$ be the *shortest* such extension $\mathring{\Pi}$, then $\widetilde{\Pi}\omega$ is reachable from ω. If $\widetilde{\Pi}\omega$ were non-terminal, $\tilde{\pi}\widetilde{\Pi}\omega$ would also be reachable from ω, so $\widetilde{\Pi}\omega$ is terminal. Thus $\widetilde{\Pi}\omega \in W_\omega$. ■

After having put the foundations in place by establishing enough of the structure of $\mathcal{W}_\Gamma$, it is now straight-forward to find the terms corresponding to equivalence classes of path expressions. Since such terms will introduce *new* variables, i.e. variables that do not occur in Γ, some way to select those variables which ensures that they do not clash with any existing variables in required. That is accomplished by the map ι_Γ below. Instead of trying to prevent clashes by excluding the variables which occur in Γ, however, ι_Γ simply assigns distinct variables to *all* terminal equivalence classes, and Θ_Γ hence renames *all* the variables in Γ, thereby avoiding any possibility of conflicts.

Definition 2.4.12. *For a bounded set of path equations* Γ*, let*

(i) $\iota_\Gamma : \dot{\mathcal{W}}_\Gamma \to \mathcal{V}$ *be an embedding of* $\dot{\mathcal{W}}$ *into* $\mathcal{V}$*,*

(ii) $\Theta_\Gamma : \mathcal{W}_\Gamma \to \mathcal{E}$ *be the recursively defined map from equivalence classes to term*

$$\Theta_\Gamma(\omega) := \begin{cases} \iota_\Gamma(\omega) & \text{if } \omega \text{ is terminal,} \\ \Theta_\Gamma(L\omega) \dot{\rightarrow} \Theta_\Gamma(R\omega) & \text{otherwise.} \end{cases}$$

Proof that Θ_Γ *is well-defined.* The bound M^ω_Γ from lemma 2.4.10 ensures that the recursive expansion of $\Theta_\Gamma(\omega)$ reaches the first case, and thus stops expanding, no later than at the M^ω_Γ-th level. $\Theta_\Gamma(\omega)$ is thus a term in $\mathcal{E}$ and depth $\Theta_\Gamma(\omega) \leq M^\omega_\Gamma$. ■

The crucial property of the Θ_Γ defined above is that the terms corresponding to path expressions mimick the structure of $\mathcal{W}$, i.e. that e.g. the term corresponding to $L\omega$ is indeed the left subterm of the term corresponding to ω. This fact is formalized by

Lemma 2.4.13. *For a bounded set of path equations* Γ *and* $\omega \in \mathcal{W}_\Gamma$,

$$\Pi\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma\big(\Pi\omega\big) \neq \bot \quad \text{if } \Pi\omega \text{ is reachable from } \omega.$$

Proof. If ω is non-terminal, then by definition $\Theta_\Gamma(\omega) = \Theta_\Gamma(L\omega) \dot{\rightarrow} \Theta_\Gamma(R\omega)$, and it follows that $\pi\Theta_\Gamma(\omega) = \Theta_\Gamma(\pi\omega)$ for $\pi \in \{L, R\}$. The general case follows by induction on the length of Π, and $\Pi\omega$ being reachable from ω guarantees that all induction steps encounter non-terminal equivalence classes.

If ω is terminal, $\Pi\omega$ is only reachable for $\Pi = \varepsilon$ and the assertion is trivially true in this case. ■

It remains to explicitly define the substitutions S and $S_1, S_2, \ldots$ using the terms Θ_Γ. S is particularly simply – each variable is simply replaced by the term corresponding to its equivalence class. Note that this includes *all* the variables that appear in Γ, even those which aren't expanded into composite terms, but rather just mapped to variables again. S thus always renames all variables.

One could, in principle, define S_i similarly by stating that $S_i\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma(\omega S_i)$. But that overdefines S_i, because it defines not only the images of *variables* under S_i, but the images of *arbitrary* terms. So if one chooses that route (which is certainly possible!) one would need to then prove that

all these definitions are in fact compatible. So instead, the following defines only the images of *variables*. But note that for those, the definition below is identical to the one that was just given! Assume that ω is terminal, i.e. that $\sigma = \Theta_\Gamma(\omega) = \iota_\Gamma(\omega) \in \mathcal{V}$. Then $S_i : \sigma \to \Theta_\Gamma\big((\iota_\Gamma^{-1}(\sigma))S_i\big)$ is just a more explicit way of expressing $S_i\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma(\omega S_i)$.

Theorem 2.4.14. *For a bounded set of path equations Γ and Θ_Γ as in 2.4.12, let*

$$S : \mathcal{V} \to \mathcal{E} : \mu \to \Theta_\Gamma\Big(\big[\mu\big]_\Gamma\Big),$$

$$S_i : \iota_\Gamma\big(\dot{\mathcal{V}}_\Gamma\big) \to \mathcal{E} : \sigma \to \Theta_\Gamma\Big(\big(\iota_\Gamma^{-1}(\sigma)\big)S_i\Big)$$

Then

(i) $S_i\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma(\omega S_i) \neq \bot$

(ii) $\Pi\Big(\big(\Theta_\Gamma(\omega)\big)\Sigma\Big) = \Theta_\Gamma\big(\Pi\omega\Sigma\big) \neq \bot$ *if* $\Pi\omega\Sigma$ *is reachable from* $\omega\Sigma$,

(iii) $S, S_1, \ldots$ *form a solution of* Γ.

Proof.

(i). (a). If ω is terminal, $\Theta_\Gamma(\omega) = \iota_\Gamma(\omega) = \sigma \in \iota_\Gamma\big(\dot{\mathcal{V}}_\Gamma\big) \subset \mathcal{V}$, and

$$S_i\big(\Theta_\Gamma(\omega)\big) = S_i\big(\iota_\Gamma(\omega)\big) = S_i\big(\sigma\big) = \Theta_\Gamma\Big(\big(\iota_\Gamma^{-1}(\sigma)\big)S_i\Big) = \Theta_\Gamma(\omega S_i) \neq \bot$$

follows from the definition of S_i.

(b). Otherwise, ω is non-terminal. Let W_ω be as in lemma 2.4.11 and $\tilde{\pi}\widetilde{\Pi} \in W_\omega$. $\tilde{\pi}\widetilde{\Pi}\omega$ is then terminal and case (a) thus applies to it, justifying ⓐ. Since $\tilde{\pi}\widetilde{\Pi}\omega$ is reachable from ω, $\widetilde{\Pi}\omega$ on the other hand is non-terminal, so is therefore $\widetilde{\Pi}\omega S_i$ due to lemma 2.4.8, and via lemma 2.4.13 this proves ①. ② follows from lemma 2.2.4, and all this together shows that

$$\tilde{\pi}\widetilde{\Pi}\Big(\Theta_\Gamma\big(\omega S_i\big)\Big) \overset{①}{=} \Theta_\Gamma\big(\tilde{\pi}\overbrace{\widetilde{\Pi}\omega S_i}^{\text{non-terminal}}\big) \overset{②}{=} S_i\Big(\Theta_\Gamma\big(\underbrace{\tilde{\pi}\overbrace{\widetilde{\Pi}\omega}^{\text{non-terminal}}}_{\text{terminal}}\big)\Big) \overset{①}{=} S_i\Big(\underbrace{\tilde{\pi}\widetilde{\Pi}\Theta_\Gamma(\omega)}_{\substack{\in\mathcal{V}\\ \neq\bot}}\Big)$$

$$\overset{②}{=} \tilde{\pi}\widetilde{\Pi}\Big(S_i\big(\Theta_\Gamma(\omega)\big)\Big) \neq \bot \qquad \text{for all } \tilde{\pi}\widetilde{\Pi} \in W_\omega.$$

By lemma 2.2.12 (note that 2.4.11 guarantees that W_ω has the required properties) this is sufficient to conclude (i).

(ii). Applying (i) repeatedly shows that $\big(\Theta_\Gamma(\omega)\big)\Sigma = \Theta_\Gamma(\omega\Sigma)$ and the rest follows from lemma 2.4.13.

(iii). Let $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'$ be an arbitrary path equations in Γ. The interpretation of that equation under $S, S_1, \ldots$ is

$$\Pi\big((\mu S)\Sigma\big) = \Pi'\big((\nu S)\Sigma'\big),$$

which after expanding the definition of S reads

$$\Pi\Big(\big(\Theta_\Gamma\big[\mu\big]_\Gamma\big)\Sigma\Big) = \Pi'\Big(\big(\Theta_\Gamma\big[\nu\big]_\Gamma\big)\Sigma'\Big).$$

Now (ii) transforms to (note that since $\Pi\mu\Sigma,\ \Pi'\nu\Sigma' \in \Delta_\Gamma$, the equivalence class $\big[\Pi\mu\Sigma\big]_\Gamma$ respectively $\big[\Pi'\nu\Sigma'\big]_\Gamma$ is reachable from $\big[\mu\Sigma\big]_\Gamma$ respectively $\big[\nu'\Sigma'\big]_\Gamma$)

$$\Theta_\Gamma\big[\Pi\mu\Sigma\big]_\Gamma = \Theta_\Gamma\big[\Pi'\nu\Sigma'\big]_\Gamma \neq \bot,$$

and that is trivially true, since $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma}$, hence $\Pi\mu\Sigma \sim_\Gamma \Pi'\nu\Sigma'$ and therefore $\big[\Pi\mu\Sigma\big]_\Gamma = \big[\Pi'\nu\Sigma'\big]_\Gamma$. ■

This finally establishes the equivalency of boundedness and solvability of set of path equations – the remaining half of the equivalency now follows trivially from the above.

Theorem 2.4.15. *A set of path equations Γ has a solution iff it is bounded.*

Proof. Follows immediately from theorem 2.3.5 and 2.4.14. ■

Corollary 2.4.16. *A (semi-)unification problem $\mathcal{S}$ has a solution iff it is bounded.*

Proof. By definition, $\mathcal{S}$ is bounded iff the corresponding set of path equations $\Gamma_{\mathcal{S}}$ is bounded. Since $\Gamma_{\mathcal{S}}$ has a solution iff it is bounded, the same holds (due to theorems 2.2.10 respectively 2.2.9) for $\mathcal{S}$. ■

An Example of How the Construction Proceeds

The following semi-unification problem

$$a \rightharpoonup b \leqq c,$$
$$b \rightharpoonup (a \rightharpoonup a) \leqq b \rightharpoonup b,$$

has the corresponding set of path equations Γ

$$aS_1 \doteq Lc \quad (E_1), \qquad bS_1 \doteq Rc \quad (E_2),$$
$$bS_2 \doteq b \quad (E_3), \qquad aS_2 \doteq Lb \quad (E_4), \qquad aS_2 \doteq Rb \quad (E_5).$$

To find S as in theorem 2.4.14, one must find all the equivalence classes reachable from $[a]_\Gamma$, $[b]_\Gamma$ or $[c]_\Gamma$. The deduction rules from 2.3.1 allow these derivations to be made (where applications of Symmetricity are implicit).

$$\dfrac{\dfrac{E_2 \quad \dfrac{E_4}{Lb \doteq aS_2}}{[LbS_1 \doteq aS_2S_1]}\,S_1}{LRc \doteq LbS_1}\,L \;\downarrow\; E_6$$

$$\dfrac{\dfrac{E_2 \quad \dfrac{E_5}{Rb \doteq aS_2}}{[RbS_1 \doteq aS_2S_1]}\,S_1}{RRc \doteq RbS_1}\,R \;\downarrow\; E_7$$

$$\dfrac{E_4 \quad E_5}{Lb \doteq Rb} \;\downarrow\; E_8$$

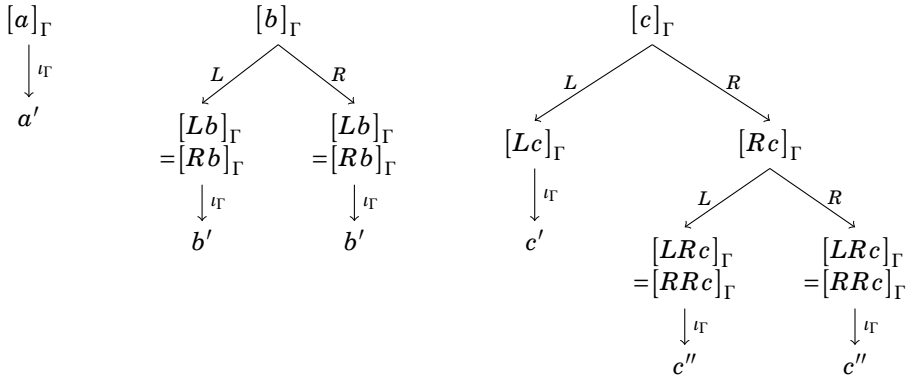
$$\dfrac{E_1 \quad \dfrac{E_4}{[Lb \doteq aS_2]}}{LbS_2 \doteq Lb}\,L \;\downarrow\; E_{10}$$

$$\dfrac{\dfrac{E_6 \quad \dfrac{E_8}{Lb \doteq Rb}}{LRc \doteq RbS_1}\,S_1 \qquad \dfrac{E_7}{RbS_1 \doteq RRc}}{LRc \doteq RRc} \;\downarrow\; E_9$$

No further L-Application or R-Application rules can be applied that would yield anything new. Applying further S_i-Application rules would, but none of the equivalence classes found that way would be reachable from $[a]_\Gamma$, $[b]_\Gamma$ or $[c]_\Gamma$. The relevant equivalence classes are thus

$$[a]_\Gamma = \{a\} \quad \text{(terminal)}$$
$$[b]_\Gamma = \{b, bS_2\} \quad \text{(non-terminal)}$$
$$[Lb]_\Gamma = \{Lb, Rb, LbS_2, aS_2\} = [Rb]_\Gamma \quad \text{(terminal)}$$
$$[c]_\Gamma = \{c\} \quad \text{(non-terminal)}$$
$$[Lc]_\Gamma = \{Lc, aS_1\} \quad \text{(terminal)}$$
$$[Rc]_\Gamma = \{Rc, bS_1\} \quad \text{(non-terminal)}$$
$$[LRc]_\Gamma = \{LRc, RRc, LbS_1, RbS_1, aS_2S_1\} = [RRc]_\Gamma \quad \text{(terminal)}$$

The following diagram shows the structure induced by L and R on these equivalence classes and the (arbitrarily named) *variables* assigned to the terminal equivalence classes by ι_Γ.



The terms assigned to $[a]_\Gamma$, $[b]_\Gamma$ and $[c]_\Gamma$ under the Θ_Γ from definition 2.4.12 follow immediately, and since those are the images of a, b and c under S, one finds that

$$S(a) = a', \quad S(b) = b' \dashrightarrow b', \quad S(c) = c' \dashrightarrow (c'' \dashrightarrow c'')$$

To find the images of a', b', c' and c'' under S_1 and S_2 as in theorem 2.4.14, one additionally needs to find the terms $\Theta_\Gamma\big(\iota_\Gamma^{-1}(\sigma)\big)$ for $\sigma \in \{a', b', c', c''\}$. One can save a bit of work by observing that c' and c'' only appear in $S(c)$, and since c doesn't appear on any left-hand side in the original semi-unification problem, the images of c' and c'' under the S_i aren't really relevant. That leaves

$$\begin{aligned} [a]_\Gamma S_1 &= [Lc]_\Gamma \\ [Lb]_\Gamma S_1 &= [LRc]_\Gamma \\ [a]_\Gamma S_2 &= [Lb]_\Gamma \\ [Lb]_\Gamma S_2 &= [Lb]_\Gamma \end{aligned}$$

and it follows that

$$S_1(a') = c', \quad S_1(b') = c'', \quad S_2(a') = b', \quad S_2(b') = b'.$$

Applying S to the original semi-unification problem yields

$$\begin{aligned} a' \dashrightarrow (b' \dashrightarrow b') &\leqq c' \dashrightarrow (c'' \dashrightarrow c''), \\ (b' \dashrightarrow b') \dashrightarrow (a' \dashrightarrow a') &\leqq (b' \dashrightarrow b') \dashrightarrow (b' \dashrightarrow b'), \end{aligned}$$

and applying S_1 to the left-hand side of the first inequality and S_2 to left-hand side of the second one shows that S, S_1, S_2 is indeed a solution because

$$\begin{aligned} c' \dashrightarrow (c'' \dashrightarrow c'') &= c' \dashrightarrow (c'' \dashrightarrow c''), \\ (b' \dashrightarrow b') \dashrightarrow (b' \dashrightarrow b') &= (b' \dashrightarrow b') \dashrightarrow (b' \dashrightarrow b'). \end{aligned}$$

2.5 Solvability of Path Equations over Infinite Trees

Perusing the construction of solutions in section 2.4, one can observe that it is only the requirement of *finiteness* imposed on terms that prevents unbounded sets of path equations from having a solution. This section studies the effects of dropping that requirement, i.e. of replacing the set of terms $\mathcal{E}$ with the set of (possibly infinite) binary trees. If this is done without any other modifications *every* set of path equations becomes solvable. In fact, there then even exists a *single* substitution S which solves every system of path equations – this, for example, holds for the substitution S which maps every variable to the infinite tree which contains *no* variables at all.

The existence of such a global unifier S for all terms makes the solvability of systems of path equations over $\overset{\infty}{\mathcal{E}}$ a rather trivial matter. Once tree are allowed to contain *constants* as well as variables, however, the problem becomes non-trivial again.

Thus, in the following, $\mathcal{C}$ will denote a infinite set of *constants*, which may appear in any place that variables where so far allowed to appear, but on which substitutions must act as identities. Since the recursive construction of $\mathcal{E}$ from definition 2.1.1 does not lend itself to an extension which includes objects of infinite depth, the following uses the definition from [JK93] instead. It defines (finite or infinite) binary trees as a sets of paths plus an assignment of variables or constants to the *maximal* paths in this set, called the *exterior* (or *leaves*) of the tree.

Definition 2.5.1 (Trees)**.** *The (uncountable!) set of all* trees *is denoted $\overset{\infty}{\mathcal{E}}$, and contains pairs $t = \langle t_{\mathcal{P}}, t_* \rangle$, where*

(i) $t_{\mathcal{P}} \subset \mathcal{P}$ is the non-empty set of defined paths *of t and must satisfy*

$$\begin{array}{rcll} L\Pi \in t_{\mathcal{P}} & \Leftrightarrow & R\Pi \in t_{\mathcal{P}} & \text{for all } \Pi \in \mathcal{P}, \\ \Pi \in t_{\mathcal{P}} & \Rightarrow & \Pi' \in t_{\mathcal{P}} & \text{for all } \Pi, \Pi' \in \mathcal{P} \text{ with } \Pi' \leq \Pi, \end{array}$$

(ii) t_* is the variable and constant assignment *of t and is a function*

$$t_* : \operatorname{ext} t \to \mathcal{V} \cup \mathcal{C}, \qquad \text{where} \qquad \operatorname{ext} t := \left\{ \Pi \in t_{\mathcal{P}} \;\middle|\; \widetilde{\Pi}\Pi \notin t_{\mathcal{P}} \text{ for } \widetilde{\Pi} \neq \varepsilon \right\}$$

In the definition above, each path in $t_{\mathcal{P}}$ "names" a node in the tree t. An equivalent, but sometimes more convenient, definition of trees as special kinds of (possibly infinite) graphs allows for arbitrary "names" of *interior* nodes, and represents the tree's structure via functions t_L, t_R which map each node to its left respectively right subtree. (In the language of graph theory, a tree is thus a directed graph where each node has either zero or two outgoing edges)

Definition 2.5.2 (Graph Representation of Trees)**.** *A $t \in \overset{\infty}{\mathcal{E}}$ is represented by a quadruple $\langle t_{\mathcal{I}}, t_\top, t_L, t_R \rangle$ consisting of a countable set $t_{\mathcal{I}}$ of* nodes *which do not include $\bot$ and are disjoint from $\mathcal{V}$ and $\mathcal{C}$, a* root *$t_\top \in t_{\mathcal{I}} \cup \mathcal{V} \cup \mathcal{C}$ and functions*

$$t_L, t_R : t_{\mathcal{I}} \to t_{\mathcal{I}} \cup \mathcal{V} \cup \mathcal{C}.$$

For $\Pi = \pi_n \dots \pi_1 \in \mathcal{P}$, t_Π again means $t_{\pi_n} \dots t_{\pi_1}$.

To translate from the former definition to the latter, one simply sets

$$t_\top := \begin{cases} \varepsilon & \text{if } \varepsilon \notin \operatorname{ext} t, \\ t_*(\varepsilon) & \text{if } \varepsilon \in \operatorname{ext} t, \end{cases}, \qquad t_{\mathcal{I}} := t_{\mathcal{P}},$$

$$t_L : \Pi \mapsto \begin{cases} L\Pi & \text{if } \Pi \notin \operatorname{ext} t, \\ t_*(\Pi) & \text{if } \Pi \in \operatorname{ext} t, \end{cases} \qquad t_R : \Pi \mapsto \begin{cases} R\Pi & \text{if } \Pi \notin \operatorname{ext} t, \\ t_*(\Pi) & \text{if } \Pi \in \operatorname{ext} t. \end{cases}$$

For the other direction, one similarly sets

$$t_{\mathcal{P}} := \left\{ \Pi \in \mathcal{P} \;\middle|\; t_\Pi(t_\top) \neq \bot \right\}, \qquad t_* : \Pi \to t_\Pi(t_\top).$$

The *complete* tree τ is simplest infinite tree in $\overset{\infty}{\mathcal{E}}$. It defines *all* possible paths, and hence contains *no* variables. Just as for terms, the symbol $\bot$ again denotes *undefined trees*.

Definition 2.5.3 (The Undefined and the Complete Infinite Trees)**.**

(i) The complete tree *$\tau \in \overset{\infty}{\mathcal{E}}$ is the infinite tree with $\tau_{\mathcal{P}} = \mathcal{P}$. Its exterior* ext *$\tau$ is the empty set, and τ hence contains* no *variables.*

(ii) The symbol $\bot$ denotes the special undefined tree*, and $\bot_{\mathcal{P}} = \bot_{\mathcal{I}} = \emptyset$. As for terms, $\bot \notin \overset{\infty}{\mathcal{E}}$, and $\overset{\infty}{\mathcal{E}}{}^{\bot} := \overset{\infty}{\mathcal{E}} \cup \{\bot\}$.*

The tree τ highlights the main downside of definition 2.5.2 – the representation of trees as quadruples $\langle t_\mathcal{I}, t_\top, t_L, t_R \rangle$ is *not unique*. For example, τ is represented by both

$$\tau_\top = 1 \qquad \tau_\mathcal{I} = \{\tau_\top\} \qquad \tau_L(t_\top) = \tau_R(\tau_\top) = t_\top$$

as well as

$$\tau_\top = 1 \qquad \tau_\mathcal{I} = \{\tau_\top, 2\} \qquad \tau_L(\tau_\top) = \tau_R(\tau_\top) = 2, \quad \tau_L(2) = \tau_R(2) = \tau_\top.$$

Matters regarding tree *equality* thus always refer to the representation from definition 2.5.1.

The depth of a tree, just as the depth of a term, is the length of the longest path that leads to a variable or constant.

Definition 2.5.4 (Depth of a Tree)**.** *The* depth *of a tree $t \in \overset{\infty}{\mathcal{E}}$ is*

$$\operatorname{depth} t := \max_{\Pi \in t_\mathcal{P}} |\Pi|.$$

In the interpretation of definition 2.5.2, the definitions of the functions[1] L and R are quite trivial – that interpretation already provides suitable functions t_L and t_R, so all that L and R do is to *redefine* $t_\top$ to refer to the new root. For clarity, the following definition also provides a definition of these functions which operates on the interpretation of trees as pairs $\langle t_\mathcal{P}, t_* \rangle$. Comparing these two definitions to the translations between these interpretations outlined above shows that these two version of the definition do indeed yield identical functions on $\overset{\infty}{\mathcal{E}}$.

Definition 2.5.5. *The definitions of the functions L,R on $\overset{\infty}{\mathcal{E}}$ are*

(i) For the interpretation of trees $t \in \overset{\infty}{\mathcal{E}}$ as quadruples $t = \langle t_\mathcal{I}, t_\top, t_L, t_R \rangle$

$$L : \overset{\infty}{\mathcal{E}} \to \overset{\infty}{\mathcal{E}}^\bot : \langle t_\mathcal{I}, t_\top, t_L, t_R \rangle \mapsto \begin{cases} \langle t_\mathcal{I}, t_L(t_\top), t_L, t_R \rangle & \text{if } t_\top \in t_\mathcal{I}, \\ \bot & \text{otherwise,} \end{cases}$$

$$R : \overset{\infty}{\mathcal{E}} \to \overset{\infty}{\mathcal{E}}^\bot : \langle t_\mathcal{I}, t_\top, t_L, t_R \rangle \mapsto \begin{cases} \langle t_\mathcal{I}, t_R(t_\top), t_L, t_R \rangle & \text{if } t_\top \in t_\mathcal{I}, \\ \bot & \text{otherwise,} \end{cases}$$

(ii) For the interpretation of trees $t \in \overset{\infty}{\mathcal{E}}$ as pairs $t = \langle t_\mathcal{P}, t_* \rangle$

$$L : \overset{\infty}{\mathcal{E}} \to \overset{\infty}{\mathcal{E}}^\bot : \langle t_\mathcal{P}, t_* \rangle \mapsto \langle \{\, \Pi \mid \Pi L \in t_\mathcal{P} \,\}, \Pi \mapsto t_*(\Pi L) \rangle,$$
$$R : \overset{\infty}{\mathcal{E}} \to \overset{\infty}{\mathcal{E}}^\bot : \langle t_\mathcal{P}, t_* \rangle \mapsto \langle \{\, \Pi \mid \Pi R \in t_\mathcal{P} \,\}, \Pi \mapsto t_*(\Pi R) \rangle,$$

As for terms, if $\Pi = \pi_n \dots \pi_1 \in \mathcal{P}$ then Πt means $\pi_1 \dots \pi_1(t)$.

For finite trees, lemma 2.2.14 showed that the set of paths leading to a variable have the "witness property", i.e. that every other path in $\mathcal{P}$ is either an extension or a prefix of one of these paths. For infinite trees, this is no longer true – in the complete tree τ, for example, *no* paths lead to variables (since there are none). But τ is, in a way, the *only* tree with this property, as the following lemma shows. Any path that is neither a prefix, nor an extension, of a path that leads to a variable must necessarily point to a subtree equal to τ.

Lemma 2.5.6. *For every tree $t \in \overset{\infty}{\mathcal{E}}$ and every path $\Pi \in \mathcal{P}$, exclusively either*

(i) $\Pi' t \in \mathcal{V} \cup \mathcal{C}$ for some $\Pi' \leq \Pi$, or

(ii) $\Pi' t \in \mathcal{V} \cup \mathcal{C}$ for some $\Pi' > \Pi$, or

(iii) $\Pi t = \tau$, where τ is the complete tree.

Proof. Let t be a tree and Π a path. *Case (a).* If $\Pi t \in \mathcal{V} \cup \mathcal{C}$, (i) applies. *Case (b).* If $\Pi t = \bot$, then since $t \neq \bot$ there are prefixes $\Pi' \leq \Pi$ with $\Pi' t \neq \bot$, and for the longest such prefix $\Pi' t \in \mathcal{V} \cup \mathcal{C}$, hence (i) applies again. *Case (c).* If $\Pi' t = \bot$ for some $\Pi' > \Pi$, then according to (b) $\Pi'' t \in \mathcal{V} \cup \mathcal{C}$ for some $\Pi'' \leq \Pi' \geq \Pi$. But then necessarily $\Pi'' \geq \Pi$, and (ii) applies. *Case (d).* If neither (i) nor (ii) apply, then in particular $\Pi t \neq \bot$, and $\widetilde{\Pi}\Pi \neq \bot$ for all $\widetilde{\Pi} \in \mathcal{P}$. But that implies $\Pi t = \tau$ and thus (iii). ■

[1] Remember that these are closely related, but not identical, to the *symbols* $L,R \in \mathcal{P}$ – the latter are purely *syntactical* objects, while the former are actual functions

Two terms are, unsurprisingly, equal exactly if they contain the same variables at the same positions. The following lemma formalizes this, and will be used as a replacement for lemma 2.2.12, which as explained above fails for infinite trees.

Lemma 2.5.7. *If $t, u \in \mathring{\mathcal{E}}$ and $t \neq u$, then $\Pi t \neq \Pi u \in \mathcal{V} \cup \mathcal{C}$ for some $\Pi \in \mathcal{P}$.*

Proof. If $t \neq u$, then either $t_{\mathcal{P}} \neq u_{\mathcal{P}}$ or $t_*(\Pi) \neq u_*(\Pi)$ for some $\Pi \in \operatorname{ext} u$. In the latter case, $\Pi u \in \mathcal{V} \cup \mathcal{C}$ and $\Pi t \neq \Pi u$ follow immediately. Assume thus that $t_{\mathcal{P}} \neq u_{\mathcal{P}}$, in which case without loss of generality one may assume that for some $\Pi \in \mathcal{P}$, $\Pi \in t_{\mathcal{P}}$ but $\Pi \notin u_{\mathcal{P}}$. But then for some prefix $\Pi' < \Pi$, $\Pi' \in u_{\mathcal{P}}$ and for the longest such prefix, $\Pi' \in \operatorname{ext} u$ and thus $\Pi u \in \mathcal{V} \cup \mathcal{C}$. Since $\Pi' < \Pi$, and since $\Pi \in t_{\mathcal{P}}$, $\Pi' \notin \operatorname{ext} t$ and thus $\Pi' t = \bot \neq \Pi' u \in \mathcal{V} \cup \mathcal{C}$ as required. ■

The simplest class of trees in $\mathring{\mathcal{E}}$ are the *finite* trees, which contain only finitely many nodes and correspond to terms in $\mathcal{E}$ except that they may include constants. A larger class is formed by the *rational* trees[1], which may have infinitely many nodes, but only finitely many non-identical subtrees. Put differently, rational trees only contains infinitely many nodes if one insists (as definition 2.5.1 does, but 2.5.2 doesn't) that distinct paths leads to distinct nodes, even if those nodes represent *identical* subterms. But if one allows nodes to be "reused", then rational trees can be represented using only a *finite* set of nodes.

Definition 2.5.8 (Classes of Trees)**.** *The set of trees $\mathring{\mathcal{E}}$ contains the subsets*

(i) $\mathcal{E}$, containing constant-free trees t with $|t_{\mathcal{P}}| < \infty$, called terms[2]*,*

(ii) $\hat{\mathcal{E}}$, containing trees t with $|t_{\mathcal{P}}| < \infty$, called finite trees *or* terms with constants*,*

(iii) $\mathring{\mathcal{E}}$, containing trees t with $|t_{\mathcal{I}}| < \infty$, called rational trees[3]*.*

Substitutions on $\mathring{\mathcal{E}}$ are characterized by turning the property *proven* for substitutions on $\mathcal{E}$ in lemma 2.2.4 – namely that substitutions are precisely those maps which commute with L and R – into a *definition*. As the name *constants* implies, substitutions are of course also prevented from replacing constant symbols.

Definition 2.5.9. *A function $S : \mathring{\mathcal{E}} \to \mathring{\mathcal{E}}$ is called a* substitution *on $\mathring{\mathcal{E}}$ if*

$$S(\eta) = \eta \qquad \text{for all } \eta \in \mathcal{C},$$
$$S(\Pi t) = \Pi S(t) \neq \bot \qquad \text{for all } t \in \mathring{\mathcal{E}} \text{ and all } \Pi \in \mathcal{P} \text{ with } \Pi t \neq \bot.$$

A substitution is called finite *respectively* rational *if $S(\mu)$ is finite respectively rational for all $\mu \in \mathcal{V}$.*

Again as for terms, there is a one-to-one correspondence between arbitrary function $\mathring{S} : \mathcal{V} \to \mathring{\mathcal{E}}$ and substitutions on $\mathring{\mathcal{E}}$.

Lemma 2.5.10. *For every $\mathring{S} : \mathring{\mathcal{V}} \to \mathring{\mathcal{E}}$ with $\mathring{\mathcal{V}} \subset \mathcal{V}$ there is a unique substitution S on $\mathring{\mathcal{E}}$, called the* extension *of $\mathring{S}$ to $\mathring{\mathcal{E}}$, with $S|_{\mathring{\mathcal{V}}} = \mathring{S}$ and $S|_{\mathcal{V} \setminus \mathring{\mathcal{V}}} = \mathrm{id}$. S. Conversely, for every substitution S on $\mathring{\mathcal{E}}$, the extension of $\mathring{S} = S|_{\mathcal{V}}$ to $\mathring{\mathcal{E}}$ is again S.*

Path equations can be used to describe trees as well, but must then be allowed to contain constants as well as variables.

Definition 2.5.11 (Path Expressions and Equations with Constants)**.**

(i) The set of path expressions with constants *$\mathcal{W}$ is*

$$\mathcal{W} := \{\, \Pi\mu\Sigma \mid \mu \in \mathcal{V} \cup \mathcal{C},\ \Pi \in \mathcal{P},\ \Sigma \in \mathcal{R} \,\}$$

(ii) A path equation with constants *is an equation of the form*

$$\Pi_1\mu\Sigma_1 \doteq \Pi_2\nu\Sigma_2$$

where $\Pi_1\mu\Sigma_1,\ \Pi_2\nu\Sigma_2 \in \mathcal{W}$ are path expressions with constants*.*

---

[1] What this text calls *rational tree* is called *regular term* in [JK93]

[2] This may be read as either a *redefinition* of $\mathcal{E}$ or as an embedding of the original set $\mathcal{E}$ into $\mathring{\mathcal{E}}$

[3] Since the representation of trees as graphs, i.e. as quadruples $\langle t_{\mathcal{I}}, t_{\top}, t_L, t_R \rangle$, is non-unique, requiring that $|t_{\mathcal{I}}| < \infty$ is slightly ambiguous. What is meant is, of course, that *some* representation which satisfies this requirement must exist.

Note that $\mathcal{W}$ denotes both the original set of *path expressions* from definition 2.2.3, and the expanded version which may include constants.

With these definitions in place, the definitions of unification, semi-unification and generalized semi-unification problems can very naturally be extended to allow *solutions* over $\overset{\infty}{\mathcal{E}}$ instead of over $\mathcal{E}$. The problems themselves, however, are still only permitted to contain only *finite* trees – but these finite trees may now contain constants as well as variables. The translation into sets of path equations works as before, but course will now yield equations containing constants.

Definition 2.5.12 (Unification and Semi-Unification with Constants)**.**

(i) Unification *(2.1.4),* semi-unification *(2.1.8) and* generalized semi-unification *(2.2.15) problems with constants contain finite trees with constants instead of terms. The corresponding set of* path equations *of such a problem will likewise contain path equations with constants.*

(ii) Unifiers*,* semi-unifiers *and* local substitutions *for such problems may be arbitrary substitutions over $\overset{\infty}{\mathcal{E}}$, i.e. may introduce infinite terms.*

(iii) A solution is called finite *respectively* rational *if all substitutions $S, S_1, \ldots$ are finite respectively rational.*

As they do for problems without constants, the corresponding sets of path equations faithfully represents (semi-)unification problems with constants. Since the trees contained in such a problem are required to be *finite*, the proof from section 2.2 works unmodified (except that, again, *constants* may appear everywhere that *variables* appeared in the original proof).

Theorem 2.5.13. *Substitutions $S, S_1, \ldots$ solve a unification or semi-unification problem with constants iff they solve the corresponding set of path equations.*

For sets of path equations *without* constants, *boundedness* was shown to be necessary and sufficient for a solution over the set of *finite* terms $\mathcal{E}$ to exist. *Consistency* plays the same role for sets of path equations *with* constants, if one allows arbitrary (i.e. possibly irrational) solutions. Just as boundedness represented the requirement that terms be finite, consistency represents the requirement that solutions do not replace constants. And as for boundedness, the necessity of consistency for the existence of a solution follows almost immediately for the definition, whereas sufficiency is shown by the explicit construction of a solution.

Definition 2.5.14. *A set of path equations Γ is said to be* consistent *if*

$$\Pi\eta\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma} \text{ with } \eta \in \mathcal{C} \quad \text{implies} \quad \Pi = \varepsilon \text{ and either } \nu \notin \mathcal{C} \text{ or } \nu = \eta.$$

A semi-unification problem $\mathcal{S}$ is called consistent *if $\Gamma_{\mathcal{S}}$ is consistent.*

Theorem 2.5.15. *An inconsistent set of path equations Γ has no solution over $\overset{\infty}{\mathcal{E}}$.*

Proof. Since constants cannot contain subterms, and since every solution of a set of path equations must yield *defined* terms for all the derivable path equations (see definition 2.2.6), a set of path equations cannot have a solution if any path equation of the form $\Pi\mu\Sigma$ with $\Pi \neq \varepsilon$ and $\mu \in \mathcal{C}$ is derivable from it. Similarly, two *distinct* constants cannot be made equal by any substitution, and thus derivability of an equation of the form $\mu\Sigma \doteq \nu\Sigma'$ with $\mu, \nu \in \mathcal{C}$ and $\mu \neq \nu$ prevents a set of path equations from having a solution as well. ■

Solution of sets of path equations over the set $\overset{\infty}{\mathcal{E}}$ of infinite trees are constructed just as solutions over $\mathcal{E}$ were in section 2.4, by using the structure of $\mathcal{W}_\Gamma$ as a "template" and constructing trees alongside that structure. In the interpretation of trees as graphs from definition 2.5.2, the elements of $\mathcal{W}_\Gamma$ can even be *directly* used as the set of internal nodes, and the functions L, R on $\mathcal{W}_\Gamma$ and L, R on $\overset{\infty}{\mathcal{E}}$ are identical, except that whenever their result would be a *terminal* equivalence class, a variable or constant is returned instead.

The only remaining question is thus which terminal equivalence classes should be mapped to constants, and which to variables. Clearly, if some equivalence class contains a path expression η, with $\eta \in \mathcal{C}$, that class must be mapped to a constant. But those aren't all – since the deduction system from definition 2.3.1 wasn't modified when constants were introduced, it does not "know"

anything about constants, and treats them just as it treats variables. In particular, it does *not* allow $\eta = S_i\eta$ to be derived for arbitrary constants η, even though this path equation must obviously be satisfied by every substitution S_i. This "blindness" of the deduction system is made up for by mapping not only the equivalence classes $[\eta]_\Gamma$ for $\eta \in \mathcal{C}$ to constants, but also all equivalence classes $[\eta\Sigma]_\Gamma$ for arbitrary $\Sigma \in \mathcal{R}$.

Definition 2.5.16 (Generalization of 2.4.12)**.** *For a consistent set of path equations Γ, let*

(i) $\iota_\Gamma : \dot{\mathcal{W}}_\Gamma \to \mathcal{V}$ *be an embedding of* $\dot{\mathcal{W}}_\Gamma$ *into* $\mathcal{V}$*,*

(ii) $\Theta_\Gamma : \mathcal{W}_\Gamma \to \mathring{\mathcal{E}}$ *be the map from equivalence classes to trees defined by*

$$\Theta_\Gamma(\omega) : \omega \mapsto \langle \mathcal{W}_\Gamma, \vartheta_\Gamma(\omega), \vartheta_L, \vartheta_R \rangle$$

(in the interpretation of 2.5.2), where

$$\vartheta_L : \omega \mapsto \vartheta_\Gamma(L\omega), \qquad \vartheta_R : \omega \mapsto \vartheta_\Gamma(R\omega),$$

$$\vartheta_\Gamma(\omega) := \begin{cases} \eta & \text{if } \omega = [\eta\Sigma]_\Gamma \text{ for } \eta \in \mathcal{C},\ \Sigma \in \mathcal{R} \\ \iota_\Gamma(\omega) & \text{if } \omega \text{ is terminal and non-constant} \\ \omega & \text{otherwise.} \end{cases}$$

Proof that Θ_Γ is well-defined. The only questionable part is the case "$\vartheta_\Gamma(\omega) = \eta$ if $\omega = [\eta\Sigma]_\Gamma$ for $\eta \in \mathcal{C}$, $\Sigma \in \mathcal{R}$" in the definition of $\vartheta_\Gamma - \eta$ might not be determined *uniquely* there. But if that was the case, i.e. if there were two constants $\eta_1 \neq \eta_2$ with $[\eta_1\Sigma_1]_\Gamma = [\eta_2\Sigma_2]_\Gamma$ for some $\Sigma_1, \Sigma_2 \in \mathcal{R}$, then necessarily $\eta_1\Sigma_1 \doteq \eta_2\Sigma_2 \in \overline{\Gamma}$ would need to hold, and Γ would hence be inconsistent. ■

The paths produced by Θ_Γ again closely mimick the structure of $\mathcal{W}_\Gamma$, as they were designed to. This now pertains not only their subterm structure (i.e. part (ii) of the lemma below), but also the places at which the terms Θ_Γ contain constants.

Lemma 2.5.17 (Generalization of 2.4.13)**.** *For a consistent set of path equations Γ,*

(i) if $\eta \in \mathcal{C}$ then for arbitrary $\Sigma \in \mathcal{R}$, $[\eta\Sigma]_\Gamma$ is terminal and $\Theta_\Gamma[\eta\Sigma]_\Gamma = \eta$.

(ii) $\Pi\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma\big(\Pi\omega\big) \neq \bot$ *if $\Pi\omega$ is reachable from $\omega \in \mathcal{W}$,*

(iii) if Γ is bounded, $\Theta_\Gamma(\omega) \in \hat{\mathcal{E}}$ for all $\omega \in \mathcal{W}_\Gamma$.

(iv) if Γ is bounded and constant-free, $\Theta_\Gamma(\omega) \in \mathcal{E}$ for all $\omega \in \mathcal{W}_\Gamma$

Proof.

(i). That $\Theta_\Gamma[\eta\Sigma]_\Gamma = \eta$ follows immediately from the definition of Θ_Γ. If $[\eta\Sigma]_\Gamma$ were non-terminal, there would be a $\Pi \neq \varepsilon$ with $\Pi\eta\Sigma \in \Delta_\Gamma$, which contradicts the consistency of Γ.

(ii). By induction on the length of Π. For $\Pi = \varepsilon$, the proposition is trivially true. If $\Pi\omega$ is reachable from ω, and $\Pi = \tilde{\pi}\Pi'$, with $\tilde{\pi} \in \{L, R\}$, then $\Pi'\omega$ is reachable from ω and by the induction assumption, $\Pi'\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma\big(\Pi'\omega\big) \neq \bot$. $\Pi'\omega$ is then also necessarily non-terminal, otherwise $\Pi\omega$ would not be reachable from ω, and according to (i) $\Pi'\omega \neq [\eta\Sigma]_\Gamma$ for all $\eta \in \mathcal{C}$, $\Sigma \in \mathcal{R}$. It follows that $\vartheta_\Gamma(\omega) = \omega$, and therefore

$$\tilde{\pi}\Pi'\big(\Theta_\Gamma(\omega)\big) = \tilde{\pi}\langle \mathcal{W}_\Gamma, \Pi'\omega, \vartheta_L, \vartheta_R \rangle = \langle \mathcal{W}_\Gamma, \vartheta_\Gamma(\tilde{\pi}\Pi'\omega), \vartheta_L, \vartheta_R \rangle = \Theta_\Gamma\big(\tilde{\pi}\Pi'\omega\big) \neq \bot.$$

(iii). Follows from lemma 2.4.10, see section 2.4 for details.

(iv). Follows from (iii) plus the fact that ϑ_Γ never introduces a constant that does not appear in Γ. ■

The solutions $S, S_1, \ldots$ which solve a consistent set of path equations Γ are now again constructed from the terms produced by Θ_Γ, in exactly the same way as they were for finite solutions in section 2.4.

Theorem 2.5.18 (Generalization of 2.4.14)**.** *For a consistent set of path equations* Γ *and* Θ_Γ *as in 2.5.16, let*

$$S : \mathcal{V} \to \overset{\infty}{\mathcal{E}} : \mu \to \Theta_\Gamma\big([\mu]_\Gamma\big),$$

$$S_i : \iota_\Gamma(\dot{\mathcal{W}}_\Gamma) \to \overset{\infty}{\mathcal{E}} : \mu \to \Theta_\Gamma\big((\iota_\Gamma^{-1}(\mu))S_i\big)$$

Then

(i) $S_i\big(\Theta_\Gamma(\omega)\big) = \Theta_\Gamma(\omega S_i) \neq \bot$,

(ii) $\Pi\Big(\big(\Theta_\Gamma(\omega)\big)\Sigma\Big) = \Theta_\Gamma\big(\Pi\omega\Sigma\big) \neq \bot$ *if* $\Pi\omega\Sigma$ *is reachable from* $\omega\Sigma$,

(iii) $S, S_1, \ldots$ *form a solution over* $\overset{\infty}{\mathcal{E}}$ *of* Γ,

(iv) if Γ *is bounded,* $S, S_1, \ldots$ *form a solution over* $\hat{\mathcal{E}}$ *of* Γ.

(v) if Γ *is bounded and constant-free,* $S, S_1, \ldots$ *form a solution over* $\mathcal{E}$ *of* Γ.

Proof.

(i). $\Theta_\Gamma(\omega S_i) \neq \bot$ follows immediately from the definition of Θ_Γ. Assume thus that $S_i\big(\Theta_\Gamma(\omega)\big) \neq \Theta_\Gamma(\omega S_i)$ for some $\omega \in \mathcal{W}_\Gamma$. Then according to lemma 2.5.7 there is some path Π with $\Pi\big(S_i\big(\Theta_\Gamma(\omega)\big)\big) \neq \Pi\big(\Theta_\Gamma(\omega S_i)\big) \in \mathcal{V} \cup \mathcal{C}$. Since $\Pi\big(\Theta_\Gamma(\omega S_i)\big) \in \mathcal{V} \cup \mathcal{C}$, $\Pi\omega S_i$ must be reachable from ωS_i – otherwise $\Pi'\omega S_i$ would have to be terminal for some $\Pi' < \Pi$, therefore $\Pi'\big(\Theta_\Gamma(\omega S_i)\big) \in \mathcal{V} \cup \mathcal{C}$, and $\Pi\big(\Theta_\Gamma(\omega S_i)\big)$ would thus be undefined. And $\Pi\omega S_i$ must be terminal, again because $\Pi\big(\Theta_\Gamma(\omega S_i)\big) \in \mathcal{V} \cup \mathcal{C}$, and due to lemma 2.4.8 $\Pi\omega$ is then terminal too. Picking the longest prefix $\Pi' \leq \Pi$ of Π for which $\Pi'\omega$ is reachable from ω, and $\tilde{\Pi}$ such that $\tilde{\Pi}\Pi' = \Pi$, now leads to a contradiction, because

$$\Pi\Big(\Theta_\Gamma\big(\omega S_i\big)\Big) \overset{2.5.17}{=} \tilde{\Pi}\Theta_\Gamma\big(\overbrace{\Pi'\omega S_i}^{\text{reachable from } \omega S_i}\big) \overset{①}{=} \tilde{\Pi}\Big(S_i\big(\underbrace{\Theta_\Gamma(\overbrace{\Pi'\omega}^{\substack{\text{terminal,}\\ \text{reachable from } \omega}})}_{\in \mathcal{V}\cup\mathcal{C}}\big)\Big) \overset{2.5.17}{=} \tilde{\Pi}\Big(S_i\big(\Pi'\Theta_\Gamma(\omega)\big)\Big)$$

$$= \Pi\Big(S_i\big(\Theta_\Gamma(\omega)\big)\Big).$$

(The identity ① follows directly from the definition of S_i if $\Theta_\Gamma(\Pi'\omega) \in \mathcal{V}$. If $\Theta_\Gamma(\Pi'\omega) = \eta \in \mathcal{C}$, then by the definition of Θ_Γ, $\Pi'\omega = [\eta\Sigma]_\Gamma$ for some Σ. But then also $\Theta_\Gamma(\Pi'\omega S_i) = \Theta_\Gamma[\eta\Sigma S_i]_\Gamma = \eta$, and since $S_i(\eta) = \eta$, it follows that $S_i\big(\Theta_\Gamma(\Pi'\omega)\big) = S_i(\eta) = \eta = \Theta_\Gamma(\Pi'\omega S_i)$ as required.)

(ii). Applying (i) repeatedly shows that $\big(\Theta_\Gamma(\omega)\big)\Sigma = \Theta_\Gamma(\omega\Sigma)$ and the rest follows from lemma 2.5.17.

(iii). Let $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'$ be an arbitrary path equation in Γ. The interpretation of that equation under $S, S_1, \ldots$ is

$$\Pi\big((\mu S)\Sigma\big) = \Pi'\big((\nu S)\Sigma'\big),$$

which after expanding the definition of S reads (note that this works if μ (or ν) is a *constant* as well, because then $\mu = S(\mu) = \Theta_\Gamma[\mu]_\Gamma$)

$$\Pi\Big(\big(\Theta_\Gamma[\mu]_\Gamma\big)\Sigma\Big) = \Pi'\Big(\big(\Theta_\Gamma[\nu]_\Gamma\big)\Sigma'\Big).$$

Now (ii) transforms to (note that since $\Pi\mu\Sigma,\ \Pi'\nu\Sigma' \in \Delta_\Gamma$, the equivalence class $[\Pi\mu\Sigma]_\Gamma$ respectively $[\Pi'\nu\Sigma']_\Gamma$ is reachable from $[\mu\Sigma]_\Gamma$ respectively $[\nu'\Sigma']_\Gamma$)

$$\Theta_\Gamma[\Pi\mu\Sigma]_\Gamma = \Theta_\Gamma[\Pi'\nu\Sigma']_\Gamma \neq \bot,$$

and that is trivially true, since $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma}$, hence $\Pi\mu\Sigma \sim_\Gamma \Pi'\nu\Sigma'$ and therefore $[\Pi\mu\Sigma]_\Gamma = [\Pi'\nu\Sigma']_\Gamma$.

(iv), (v). Follows immediately from lemma 2.5.17 (iii) and (iv). ■

This proves the second half of the equivalency of consistency and solvability over $\overset{\infty}{\mathcal{E}}$, for sets of path equations with constants as well as (via theorem 2.5.13) semi-unification problems with constants.

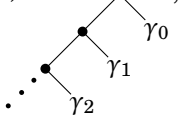
Theorem 2.5.19. *A set of path equations* Γ *with constants, respectively a semi-unification problem* $\mathcal{S}$ *with constants, has a solution over* $\overset{\infty}{\mathcal{E}}$ *iff it is consistent.* Γ *has a solution over* $\hat{\mathcal{E}}$ *iff it is bounded as well as consistent, and a solution over* $\mathcal{E}$ *iff it is bounded, consistent and constant-free.*

2.6 Solvability of Path Equations over Rational Trees

The question of whether a set of path equations Γ, or a semi-unification problem $\mathcal{S}$, have a solution over $\mathring{\mathring{\mathcal{E}}}$ is answered by theorem 2.5.19 – as that theorem shows, the existence of a solution hinges on the *consistency* of Γ respectively $\mathcal{S}$. Whether such problems have a *finite* solution, i.e. a solution over $\hat{\mathcal{E}}$, depends additionally on the *boundedness* of Γ respectively $\mathcal{S}$.

That leaves the question of solvability over $\mathring{\mathcal{E}}$. Some semi-unification problems, like for example $\mathcal{S} = \{\alpha \nrightarrow \alpha \leq \alpha\}$, clearly have a solution over $\mathring{\mathcal{E}}$ – for this problem, $S(\alpha) := \tau, S_1 = \mathrm{id}$ (where τ again denotes the complete tree) is a solution, and that is also the solution which is constructed by theorem 2.5.19. But there are also cases where theorem 2.5.19 constructs an irrational solution, yet a different solution which *does* only introduce terms in $\mathring{\mathcal{E}}$ exists. Take for example the slightly modified problem $\mathcal{S}' = \{\alpha \nrightarrow \beta \leq \alpha\}$. The associated set of path equations of this problem is $\Gamma_{\mathcal{S}'} = \{\alpha S_1 \doteq L\alpha, \beta S_1 \doteq R\alpha\}$, and playing with these path equations shows that the only really "interesting" path equations that are derivable are $\alpha S_1^n \doteq L^n \alpha$ for arbitrary $n \in \mathbb{N}$ (There are of course more derivable path equations, like $L^n \alpha S_1^m \doteq L^{n+m}\alpha$, but these follow immediately from $\alpha S_1^n \doteq L^n \alpha$). The solution produced by 2.5.19 is thus essentially

$$S(\alpha) := t^\alpha := \quad , \qquad S_1(\beta) := \gamma_0, \qquad S_1(\gamma_i) := \gamma_{i+1}.$$



Since $RL^n t^\alpha = \gamma_n$ for all n, and since all these are *different* variables, t^α is clearly irrational. But nothing in $\Gamma_{\mathcal{S}'}$ *necessitates* the introduction of all these distinct variables. Their creation is a byproduct of that fact that Θ_Γ maps two equivalence classes to the *distinct* terms unless Γ *specifically* proves them equal – but if Γ *doesn't* prove them equal, that does not imply that they *necessarily* need to be mapped to different terms to satisfy the equations in Γ. In fact, since $\mathcal{S}'$ does not contain constants, simply mapping all variables to τ again solves $\mathcal{S}'$, and *that* solution is clearly rational, since $\tau \in \mathring{\mathcal{E}}$.

Thus, so far, even though some of the solutions produced by theorem 2.5.19 were irrational, the consistent semi-unification problems which were discussed turned out to all have a solution over $\mathring{\mathcal{E}}$. This raises the question of whether this is universally true – whether every semi-unification problem, or system of path equations, which has a solution over $\mathring{\mathring{\mathcal{E}}}$ (i.e. which is consistent) also has a solution over $\mathring{\mathcal{E}}$. The answer is clearly "yes" for problems without constants – as has been mentioned multiply times, the universal unifier $S(\mu) := \tau$ for all $\mu \in \mathcal{V}$ solves *all* of these problems, without even requiring any non-trivial local substitutions.

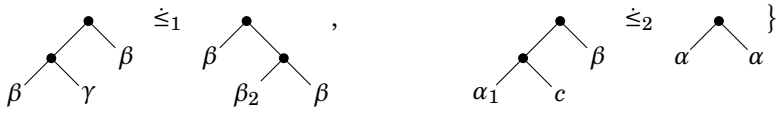
For problems with constants, however, the answer is "no". This shown now by giving an example of a semi-unification problem which has a solution over $\mathring{\mathring{\mathcal{E}}}$, but which *cannot* have a solution over $\mathring{\mathcal{E}}$ because any solution must, contrary to the case above, *necessarily* introduce terms with infinitely many distinct subterms.[1]

Theorem 2.6.1. *The existence of a solution over $\mathring{\mathring{\mathcal{E}}}$ of a system of path equations Γ is necessary, but not sufficient, for the existence of a solution over $\mathring{\mathcal{E}}$. In particular, the semi-unification problem*

$$\mathcal{S} := \{(\beta \nrightarrow \gamma) \nrightarrow \beta \leq_1 \beta \nrightarrow (\beta_2 \nrightarrow \beta), (\alpha_1 \nrightarrow c) \nrightarrow \beta \leq_2 \alpha \nrightarrow \alpha\}$$

(where $\alpha, \alpha_1, \beta, \beta_2 \in \mathcal{V}$, $c \in \mathcal{C}$), and hence also its associated system of path equations $\Gamma_{\mathcal{S}}$, have a solution over $\mathring{\mathring{\mathcal{E}}}$ but not over $\mathring{\mathcal{E}}$.

Proof. It is clear that the existence of a solution over $\mathring{\mathring{\mathcal{E}}}$ is necessary for the existence of a solution over $\mathring{\mathcal{E}}$. Reasoning about $\mathcal{S}$ and its solutions is easier if one works with a graphical representation of trees instead. Stated graphically,

$\mathcal{S} := \{$  $\}$

[1] [JK93] shows that such semi-unification problems must exist, but does so non-constructively. Finding an concrete example is mentioned as an open problem, cf. [JK93], 5.2 "Regular Semi-Unification and the Redex Procedure", page 37ff.

and $\mathcal{S}$ has the associated set of path equations

$$\Gamma_{\mathcal{S}} := \{ \quad \beta S_1 \doteq L\beta,\ \gamma S_1 \doteq R\beta,\ L\beta S_1 \doteq \beta_2,\ R\beta S_1 \doteq \beta, \\ \alpha_1 S_2 \doteq L\alpha,\ cS_2 \doteq R\alpha, \quad \beta S_2 \doteq \alpha \qquad\qquad \}.$$

To show that $\Gamma_{\mathcal{S}}$ cannot have a rational solution, the first step is to show that $R^{n+1}L^n\beta \doteq R\beta \in \overline{\Gamma}$ for all $n \in \mathbb{N}$. From $\beta S_1 \doteq L\beta$ respectively $R\beta S_1 \doteq \beta$, one can derive $L^n\beta \doteq \beta S_1^n$ respectively $R^{n+1}\beta S_1^{n+1} \doteq R^n\beta S_1^m$, both for arbitrary $n \in \mathbb{N}$, by repeatedly applying the following two proof schemes.

$$\dfrac{\dfrac{L^n\beta \doteq \beta S_1^n \qquad \dfrac{L\beta \doteq \beta S_1}{[L\beta S_1^n \doteq \beta S_1^{n+1}]}\,S_1^n}{L^{n+1}\beta \doteq L\beta S_1^n}\,L \qquad \dfrac{L\beta \doteq \beta S_1}{L\beta S_1^n \doteq \beta S_1^{n+1}}\,S_1^n}{L^{n+1}\beta \doteq \beta S_1^{n+1}}$$

$$\dfrac{\dfrac{R^{n+1}\beta S_1^{n+1} \doteq R^n\beta S_1^n}{R^{n+1}\beta S_1^{n+2} \doteq R^n\beta S_1^{n+1}}\,S_1 \qquad [R^{n+1}\beta S_1^{n+1} \doteq R^n\beta S_1^n]}{R^{n+2}\beta S_1^{n+2} \doteq R^{n+1}\beta S_1^{n+1}}\,R$$

From these path equations, $R^nL^n\beta \doteq \beta \in \overline{\Gamma}$ and finally $R^{n+1}L^n\beta \doteq R\beta \in \overline{\Gamma}$ is then derived by

$$\dfrac{\dfrac{L^n\beta \doteq \beta S_1^n \quad [R\beta S_1^n \doteq R^{n-1}\beta S_1^{n-1}]}{R^nL^n\beta \doteq R^n\beta S_1^n}\,R^n \qquad R^n\beta S_1^n \doteq R^{n-1}\beta S_1^{n-1} \ \ldots\ R\beta S_1 \doteq \beta}{\dfrac{R^nL^n\beta \doteq \beta \qquad\qquad [R\beta \doteq \gamma S_1]}{R^{n+1}L^n\beta \doteq R\beta}\,R}$$

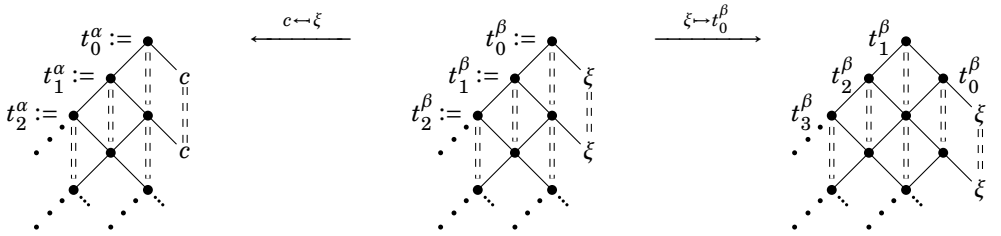
Finally, since α must be the image of β under S_2, the same structure is imposed on α. And since α's right subterm must be the constant c (per the path equation $R\alpha \doteq c$), the structure of α is in fact more rigid than that of β – while the requirements on β alone could, as usual for constant-free problems, be satisfied by τ, those one α cannot. The following derivation shows that, in particular, $R^{n+1}L^nS(\alpha)$ must be the constant c, if S is to be a semi-unifier of $\mathcal{S}$.

$$\dfrac{\dfrac{R^{n+1}L^n\beta \doteq R\beta}{R^{n+1}L^n\beta S_2 \doteq R\beta S_2}\,S_2 \qquad \dfrac{\beta S_2 \doteq \alpha \qquad [R\alpha \doteq cS_2]}{R\beta S_2 \doteq R\alpha}\,R \qquad R\alpha \doteq cS_2}{R^{n+1}L^n\beta S_2 \doteq cS_2}$$

$$\dfrac{\alpha \doteq \beta S_2 \qquad \dfrac{R^{n+1}L^n\beta \doteq R\beta}{[R^{n+1}L^n\beta S_2 \doteq R\beta S_2]}\,S_2}{\dfrac{R^{n+1}L^n\alpha \doteq R^{n+1}L^n\beta S_2}{R^{n+1}L^n\alpha \doteq cS_2}\,R^{n+1}L^n} \qquad R^{n+1}L^n\beta S_2 \doteq cS_2$$

This, however, precludes $\mathcal{S}$ having a rational solution. Since any solution of $\mathcal{S}$ must satisfy $R^{n+1}L^n\alpha = cS_2 = c$ for arbitrary $n \in \mathbb{N}$, and since a *single* subterm t can only satisfy R^nt for *one* particular $n \in \mathbb{N}$, all the subterms $L^n\alpha$ of α must exists and be different. Which, obviously, no rational term $S(\alpha)$ can satisfy.

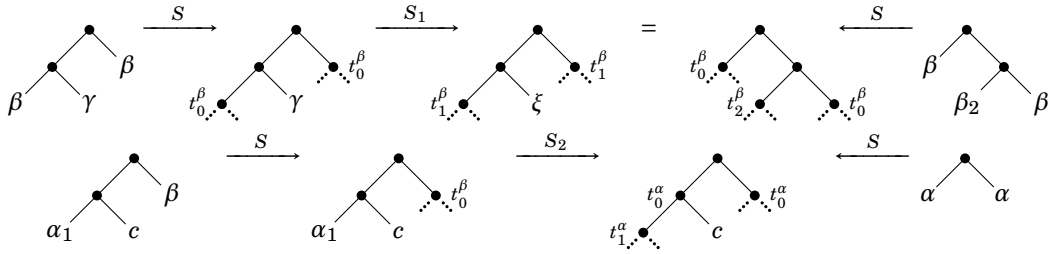
It remains to be shown that $\mathcal{S}$ *does* have a solution over $\overset{\infty}{\mathcal{E}}$. The crucial step in the construction of such a solution is to find a suitable term t^β which can serve as β's image under the global substitution S. It must, in particular, satisfy $t^\beta S_1 = Lt^\beta$, i.e. its left subterm must be its image under S_1, and $Rt^\beta S_1 = t^\beta$, i.e. its right subterm's image under S_1 must be the whole term. The following diagram presents a candidate for t^β – called t_0^β, and its (iterated) left subterms t_i^β which have the required property that t_i^β can be transformed into t_{i+1}^β by applying a suitable substitution. The diagram also shows how t_0^β is transformed into t_0^α, i.e. α's image under S. Note that all nodes within each column represent the *same* subterm, as indicated by the equal signs (=) between those nodes.



The diagram already shows all important replacements the substitutions S, S_1, S_2 must do if they are to solve $\mathcal{S}$. The full definition for these substitutions, including the images of the auxiliary variables α_1, β_2 and γ, is

$$\begin{aligned} S(\alpha) &:= t_0^\alpha, & S_1(\xi) &:= t_0^\beta, & S_2(\xi) &:= c, \\ S(\beta) &:= t_0^\beta, & S_1(\gamma) &:= \xi, & S_2(\alpha_1) &:= t_1^\alpha, \\ S(\beta_2) &:= t_2^\beta. \end{aligned}$$

It remains to verify that S, S_1, S_2 do indeed solve $\mathcal{S}$, which is again done graphically by comparing images of the left-hand sides under S and S_1 respectively S_2 which those of the right-hand sides under S.



S, S_1, S_2 thus form a solution of $\mathcal{S}$ over $\overset{\infty}{\mathcal{E}}$, which concludes the proof. ■

2.7 Principal Solutions

Solutions of systems of path equations – and thus of semi-unification problems – are in general not unique. The simple, constant-free system $\mathcal{S} = \{\alpha \leq \alpha \dashrightarrow \beta\}$ for example, has the finite solution $S = \mathrm{id}$, $S_1(\alpha) := \alpha \dashrightarrow \beta$, but also the finite solution $S'(\beta) := \alpha$, $S'_1(\alpha) := \alpha \dashrightarrow \alpha$. The second solution, however, introduces more equalities between subterms than $\mathcal{S}$ actually demands – from $\Gamma_{\mathcal{S}}$ only the path equations $\overline{\Gamma}_{\mathcal{S}} = \{L\alpha S_1 \doteq \alpha, R\alpha S_1 \doteq \beta\}$ are derivable, whereas the second solution also fulfills $R\alpha S_1 \doteq \alpha$. This introduces an asymmetry between the semi-unifiers S and S' – while S' can be expressed as $S' = \widetilde{S} \circ S$ (since $S = \mathrm{id}$, simply setting $\widetilde{S} = S'$ suffices), the reverse is not possible, since no $\widetilde{S}$ can "undo" the unification of the variables α and β done by S'. This motivates

Definition 2.7.1. *Let E be one of $\mathcal{E}$, $\hat{\mathcal{E}}$, $\mathring{\mathcal{E}}$ or $\mathring{\hat{\mathcal{E}}}$, and let $S, S_1, \ldots$ and $S', S'_1, \ldots$ be two solutions over E of some system of path equations Γ. $S, S_1, \ldots$ is called* more general *than $S', S'_1, \ldots$ iff there exist a substitution $\widetilde{S}$ over E with $S'(\mu) = \widetilde{S}(S(\mu))$ for all variable $\mu \in \mathcal{V}$ that occur in Γ.*

A solution over $E \in \{\mathcal{E}, \hat{\mathcal{E}}, \mathring{\mathcal{E}}, \mathring{\hat{\mathcal{E}}}\}$ that is more general than every other solution over E is called a most general *or* principal *solution over E of Γ.*

As the above already hints at, solutions are non-principal if they satisfy more path equations than strictly necessary. The following definition and lemma express this connection.

Definition 2.7.2. *For a substitution S and a set of variables $V \subset \mathcal{V}$, the set of path equations concerning V satisfied by S is*

$$\Gamma_S^V := \{\, \Pi\mu \doteq \Pi'\nu \mid \Pi, \Pi' \in \mathcal{P},\ \mu, \nu \in V,\ \Pi(S(\mu)) = \Pi'(S(\nu)) \neq \bot \,\}.$$

Lemma 2.7.3. *If S, S' are substitutions over $E \in \{\hat{\mathcal{E}}, \mathring{\mathcal{E}}, \mathring{\hat{\mathcal{E}}}\}$ and $V \subset \mathcal{V}$ a set of variables, then $\Gamma_S^V \subset \Gamma_{S'}^V$ iff there exists a substitution $\widetilde{S}$ over E with $S'(\mu) = \widetilde{S}(S(\mu))$ for all $\mu \in V$.*

Proof.

Assume that $S'(\mu) = \widetilde{S}(S(\mu))$ for some $\widetilde{S}$. If $\Pi\mu \doteq \Pi'\nu \in \Gamma_S^V$ then by definition $\Pi(S(\mu)) = \Pi'(S(\nu)) \neq \bot$. Applying $\widetilde{S}$ to both sides and moving the applications of Π and Π' outwards yields $\Pi(\widetilde{S}(S(\mu))) = \Pi'(\widetilde{S}(S(\nu))) = \bot$ and hence $\Pi\mu \doteq \Pi'\nu \in \Gamma_{S'}^V$.

Assume that $\Gamma_S^V \subset \Gamma_{S'}^V$. If $\Pi\mu S(\mu) \neq \bot$ then $\Pi\mu \doteq \Pi\mu \in \Gamma_S^V \subset \Gamma_{S'}^V$, and hence $\Pi\mu S'(\mu) \neq \bot$. Also due to $\Gamma_S^V \subset \Gamma_{S'}^V$, if $\Pi S(\mu) = \Pi' S(\nu)$ then $\Pi S'(\mu) = \Pi' S'(\nu)$ as well. These two results allow a substitution $\widetilde{S}$ to be defined unambiguously as

$$\widetilde{S}(\sigma) := \widetilde{\Pi} S'(\mu) \quad \text{if } \widetilde{\Pi} S(\mu) = \sigma \text{ for some } \mu \in V.$$

and this substitution then satisfies $S'(\mu) = \widetilde{S}(S(\mu))$ for all $\mu \in V$. ■

The solution constructed by 2.5.18 are minimal in the sense of satisfying only the path equations that are absolutely necessary. The proof of the following theorem exploits that, plus the result above, to show that these solution are always *principal* solutions.

Theorem 2.7.4. *For every consistent system of path equations Γ, the solution constructed by theorem 2.5.18 is principal over $\mathring{\hat{\mathcal{E}}}$.*

Proof. In the light of lemma 2.7.3, it suffices to show that for ever solution S' of Γ over $\mathring{\hat{\mathcal{E}}}$, $\Gamma_{S'}^V \supset \Gamma_S^V$, where S is the solution constructed by theorem 2.5.18, and V the variables occurring in Γ. Let thus $\Pi\mu \doteq \Pi'\nu$ be a path equation in Γ_S^V, i.e. assume that $\Pi(S(\mu)) = \Pi'(S(\nu)) \neq \bot$, and let $\widetilde{\Pi} \in \mathcal{P}$ be a path with $\widetilde{\Pi}\Pi(S(\mu)) = \widetilde{\Pi}\Pi'(S(\nu)) \in \mathcal{V} \cup \mathcal{C}$. The following case distinction shows that then $\widetilde{\Pi}\Pi S'(\mu) = \widetilde{\Pi}\Pi S'(\nu)$ holds whenever S' is a solution of Γ, and together with lemma 2.5.7 this yields $\Pi\mu \doteq \Pi'\nu \in \Gamma_{S'}^V$ for every solution S' of Γ.

(a) $\widetilde{\Pi}\Pi S(\mu) = \widetilde{\Pi}\Pi' S(\nu) = \eta \in \mathcal{C}$. According to the definition 2.5.16 of Θ_Γ, this situation only occurs if $\widetilde{\Pi}\Pi\mu \doteq \eta\Sigma_1$ and $\widetilde{\Pi}\Pi'\nu \doteq \eta\Sigma_2$ are derivable from Γ. But then $\widetilde{\Pi}\Pi S'(\mu) = \widetilde{\Pi}\Pi S'(\nu) = \eta$ must also hold, if S' is to be a solution of Γ.

(b) $\Pi = \Pi'$ *and* $\mu = \nu$. The only property to show here is that $\Pi S'(\mu) \neq \bot$ if S' is to solve Γ. If $\Pi\mu \in \overline{\Gamma}$, this is guaranteed by lemma 2.3.4. If $\Pi\mu \notin \Delta_\Gamma$, then $[\Pi\mu]_\Gamma$ is terminal, so for $\Pi\Theta_\Gamma(\mu) = \Pi S(\mu)$ to be $\neq \bot$, $[\Pi\mu]_\Gamma$ must be reachable from $[\mu]_\Gamma$, meaning $[\mathring{\Pi}\mu]_\Gamma$ must be non-terminal for the longest

true prefix $\mathring{\Pi}$ of Π (i.e. $\Pi = \pi\mathring{\Pi}$ where $\pi \in \{L, R\}$). Thus, $\hat{\Pi}\hat{\Pi}\mathring{\Pi}\mu \in \Delta_\Gamma$ for some $\hat{\Pi} \neq \varepsilon$, and thus if S' is a solution of Γ, $\hat{\Pi}\hat{\Pi}\mathring{\Pi}S'(\mu) \neq \bot$. That in particular implies that $\mathring{\Pi}S'(\mu)$ must be a *composite* term (not a variable or constant), and it follows that $\Pi S'(\mu) = \pi\mathring{\Pi}S'(\mu) \neq \bot$.

(c) $\widetilde{\Pi}\Pi \neq \widetilde{\Pi}\Pi'$ or $\mu \neq \nu$. Since ι_Γ assigns different variables to distinct equivalence classes (see definition 2.5.16), for $\widetilde{\Pi}\Pi(S(\mu)) = \widetilde{\Pi}\Pi'(S(\nu))$ to hold, it must be that $\left[\widetilde{\Pi}\Pi\mu\right]_\Gamma = \left[\widetilde{\Pi}\Pi'\nu\right]_\Gamma$. But then, according definition 2.4.1 of $\sim_\Gamma$, there is $\hat{\Pi}\mu \doteq \check{\Pi}\nu \in \overline{\Gamma}$ with $\mathring{\Pi}\hat{\Pi} = \widetilde{\Pi}\Pi\mu$ and $\mathring{\Pi}\check{\Pi} = \widetilde{\Pi}\Pi'$. Any solution S' of Γ must thus satisfy $\hat{\Pi}S'(\mu) = \check{\Pi}S'(\nu)$. It follows from (b) that $\widetilde{\Pi}\Pi S'(\mu) \neq \bot$ for every solution S' of Γ, and since $\hat{\Pi} \leq \widetilde{\Pi}\Pi$, every solution S' also obeys $\widetilde{\Pi}\Pi S'(\mu) = \widetilde{\Pi}\Pi' S'(\nu) = \bot$ as required. ■

This provides a way of showing that whenever a semi-unification problem has *any* (finite) solution, it has a principal (finite) solution.

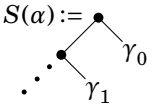
Theorem 2.7.5. *If a system or path equations respectively a semi-unification problem has a solution over $E \in \{\mathcal{E}, \hat{\mathcal{E}}, \hat{\mathcal{E}}^\infty\}$, it has a principal solution over E.*

Proof. If a system of path equations has a solution over $E \in \{\mathcal{E}, \hat{\mathcal{E}}, \hat{\mathcal{E}}^\infty\}$, per theorem 2.5.19 it must be consistent (if $E = \hat{\mathcal{E}}^\infty$), respectively consistent and bounded (if $E = \hat{\mathcal{E}}$) respectively consistent, bounded and constant-free (if $E = \mathcal{E}$). But then according to theorems 2.5.18 and 2.7.4 it has a solution in $E \subset \hat{\mathcal{E}}^\infty$ that is principal over $\hat{\mathcal{E}}^\infty$. Since a solution that is principal over $\hat{\mathcal{E}}^\infty$ is also principal over every subset of $\hat{\mathcal{E}}^\infty$, and thus in particular over E, the assertion follows. ■

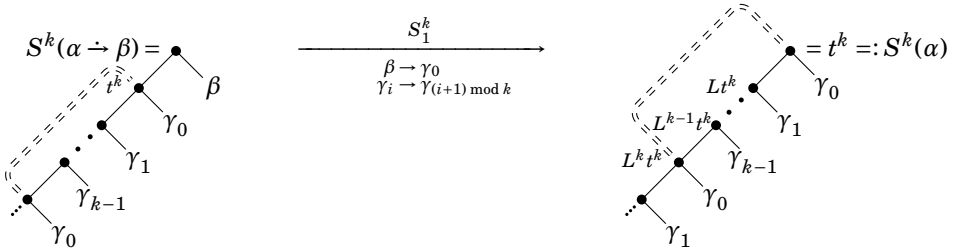
This brings up the question of whether the same is true for solutions over $\mathring{\mathcal{E}}$, i.e. if whenever a system of path equations has a solution over $\mathring{\mathcal{E}}$, it has a principal solution over $\mathring{\mathcal{E}}$. The following theorem shows that the answer is, contrary to cases of $\mathcal{E}$,$\hat{\mathcal{E}}$ and $\hat{\mathcal{E}}^\infty$, "no"[1].

Theorem 2.7.6. *The semi-unification problem $\mathcal{S} := \{\alpha \dot{\rightarrow} \beta \leq \alpha\}$, and thus $\Gamma_\mathcal{S}$, has a solution over $\mathring{\mathcal{E}}$, but no principal solution over $\mathring{\mathcal{E}}$.*

Proof. Since $\mathcal{S}$ is constant-free, $\mathcal{S}$ is trivially consistent, and theorem 2.5.18 thus constructs a solution over $\hat{\mathcal{E}}^\infty$, which is

$S(\alpha) :=$  , $S_1(\beta) := \gamma_0$, $S_1(\gamma_i) := \gamma_{i+1}$.

This solution, however, is clearly not rational, since the subtrees $L^n S(\alpha) = L^n t^\alpha$ for $n \in \mathbb{N}$ are all different. But by using only finitely many variables $\{\gamma_0, \ldots, \gamma_{k-1}\}$ instead of infinitely many variables $\{\gamma_0, \gamma_1, \ldots\}$ to build $S(\alpha)$, one can construct rational solutions of $\mathcal{S}$. The following diagram illustrates a family $\left(S^k, S_1^k\right)$ of such solutions, which arise from S by mapping γ_i to $\gamma_{i \bmod k}$. It is obvious from the picture that $S_1^k\left(S^k(\alpha \dot{\rightarrow} \beta)\right) = S^k(\alpha)$, and that all the pairs $\left(S^k, S_1^k\right)$ thus indeed from solutions of $\mathcal{S}$. ("=" again connects equal subterms)



Observe now that for $S^k(\alpha)$, k is the smallest integer for which $L^k S(\alpha) = S(\alpha)$, or in other words that $\left|\left\{ L^n S^k(\alpha) \;\middle|\; n \in \mathbb{N} \right\}\right| = k$. That, however, precludes $\mathcal{S}$ having a principal solution over $\mathring{\mathcal{E}}$! If such a principal solution $\mathring{S}, \mathring{S}_1$ existed, it would need to satisfy $\left|\left\{ L^n \mathring{S}(\alpha) \;\middle|\; n \in \mathbb{N} \right\}\right| = N < \infty$ for some N (otherwise it would not be a *rational* substitution). But then, since substitutions map equal subtrees to equal subtrees, $\left|\left\{ L^n \widetilde{S}\mathring{S}(\alpha) \;\middle|\; n \in \mathbb{N} \right\}\right| \leq N < \infty$ for every substitution $\widetilde{S}$, which contradicts the existence of a $\widetilde{S}$ with $\widetilde{S}\mathring{S}(\alpha) = S^{N+1}(\alpha)$, and hence the principality of $\mathring{S}, \mathring{S}_1$. ■

[1] This is listed as an open problem in [JK93], cf. [JK93], 7.1 "Principality Property and the Solution Set", page 48

Chapter 3

Boundedness of Turing Machines

3.1 Basic Definitions

The symbol $\mathcal{M}$ will denote a (in general non-deterministic) *Turing machine* with finite state set $\mathcal{Q}_\mathcal{M}$, and finite alphabet $\mathcal{A}_\mathcal{M}$ which includes the blank symbol $\square$. $\mathcal{M}$ is assumed to have one tape, infinite in both directions, and numbered tape cells. An *instantaneous description*, or *ID*, of $\mathcal{M}$ describes a single possible configuration of the machine. It is represented by a triple $\langle q, m, f \rangle$ where $q \in \mathcal{Q}_\mathcal{M}$ is the current state, $m \in \mathbb{Z}$ the current head position and $f : \mathbb{Z} \to \mathcal{A}_\mathcal{M}$ specifies the tape's contents. During each step, $\mathcal{M}$ transitions from one ID to a successor by overwriting the symbol at the head's position, moving the head left or right one cell and switching to a new state. The *transition relation* $\mathcal{T}_\mathcal{M} \subseteq \mathcal{Q}_\mathcal{M} \times \mathcal{A}_\mathcal{M} \times \mathcal{A}_\mathcal{M} \times \mathcal{Q}_\mathcal{M} \times \{-1, 1\}$ defines, for a combination of current state and symbol, which new symbol to write, which direction to move in and which new state to switch to.

Definition 3.1.1 (Successor Relation for Turing Machines)**.** *For a Turing machine $\mathcal{M}$ and two IDs*

$$C = \langle q, m, f \rangle,$$
$$C' = \langle q', m + \Delta, f' \rangle$$

of $\mathcal{M}$,

(i) *C' is a* successor *of C, denoted by $C \vdash_\mathcal{M} C'$, iff*

$$\langle q, f(m), f'(m), q', \Delta \rangle \in \mathcal{T}_\mathcal{M},$$
$$f'(i) = f(i) \quad \text{for all } i \neq m.$$

(ii) *C' is a n-*fold successor *of C, written $C \vdash^n_\mathcal{M} C'$, iff there are IDs $C = C_0 \vdash_\mathcal{M} C_1 \vdash_\mathcal{M} \cdots \vdash_\mathcal{M} C_n = C'$.*

(iii) *C' is* reachable *from C, written $C \vdash^\star_\mathcal{M} C'$, iff $C \vdash^n_\mathcal{M} C'$ for some n.*

If the *transition relation* contains at most one quintuple for every combination of current state and symbol, the TM $\mathcal{M}$ is called *deterministic*. In other words, a deterministic $\mathcal{M}$'s transition relation $\mathcal{T}_\mathcal{M}$ can be represented by a partial function $\mathcal{F}_\mathcal{M} : \mathcal{Q}_\mathcal{M} \times \mathcal{A}_\mathcal{M} \to \mathcal{A}_\mathcal{M} \times \mathcal{Q}_\mathcal{M} \times \{-1, 1\}$ where $\langle q, a, a', q', \Delta \rangle \in \mathcal{T}_\mathcal{M}$ iff $\mathcal{F}_\mathcal{M}(q, a) = \langle a', q', \Delta \rangle$. On deterministic machines, every ID therefore has at most one successor.

If no successor exists for an ID $C = \langle q, m, f \rangle$, that is, if no quintuple $\langle q, f(m), a', q', \Delta \rangle$ exists in $\mathcal{T}_\mathcal{M}$, C is called *final*. If $\mathcal{M}$ reaches such an ID, it *halts*.

A sequence $C_0 \vdash C_1 \vdash \cdots$ of IDs is called a *computation* if the sequence is either infinite or ends with a final ID C_n. In the former case, the length of the computation is n while in the latter it is infinite. C_0 is called the *initial ID* of the computation, and conversely the computation is said to be *initiated* by C_0 . For deterministic machines each ID initiates exactly one computation

(having length 0 if the ID is final), while for non-deterministic machines multiple computations may be initiated by the same ID. *Computation fragments* are sequences of IDs $C_0 \vdash C_1 \vdash \cdots$ which do not necessarily end with a final ID. An important property of computations and computation fragments will be the maximal distance of the head from its initial position.

Definition 3.1.2 (Tape Span)**.** *For a computation (fragment)* $C_0 \vdash_{\mathcal{M}} C_1 \vdash_{\mathcal{M}} \cdots$,

$$\operatorname*{span}_i C_i := \max_i |m_i - m_0| \quad \textit{(assuming } C_i = \langle q_i, m_i, f_i \rangle\textit{)},$$

is called the computation (fragment)'s tape span*.*

Since the head moves by only one cell between an ID and one of its successors, the tape span never exceeds the computation length. Only computation with infinite length can thus have an infinite tape span.

Since the above describes the tape's contents as an *arbitrary* function $f : \mathbb{Z} \to \mathcal{A}$, it doesn't restrict the tape to contain only finitely many non-blank symbols. The following definitions allows the distinction to be made if necessary.

Definition 3.1.3 (Finite ID)**.** *An ID* $\langle q, m, f \rangle$ *is called* finite *if* $f(n) \neq \square$ *for only finitely many* n*.*

In the following, "Turing machine" will always refer to the general non-deterministic case.

3.2 Immortality and Boundedness

Consider a particular computation initiated by some ID $C = \langle q, m, f \rangle$. That computation can then exhibit three distinct kinds of behavior

(i) First, it might reach a final ID after a finite number of steps and thus halt.

(ii) Else, it might at some point reach an ID again which it already reached before, causing it to enter an infinite loop.

(iii) And finally, the computation might run eternally without getting caught in a loop.

Note that in case (iii) the computation's tape span is necessarily infinite. Otherwise, if the head stays within a radius of n around m, the machine can reach only finitely many IDs since $f'(n) = f(n)$ for $|n - m| > R$ then holds for all reachable IDs $\langle q', m', f' \rangle$. But if it runs eternally through a finite set of IDs, it must obviously reach an ID twice.

In the following, the attention will turn to the behavior of arbitrary computations independent from their particular initial IDs. The property corresponding to (i) above is then

Definition 3.2.1 (Mortality)**.** *A Turing machine is* mortal *iff no computation with an infinite length exists,* immortal *otherwise.*

It must be stressed that mortality would be a much weaker concept if infinite IDs were disregarded. Imagine for example a Turing machine that simply moves right unless the head is positioned over a blank cell. Started from any finite ID, this machine will halt eventually. But started from an (necessarily infinite) ID without a single blank on the tape, the machine will move right forever, thus making it immortal.

The following theorem, due to Philip K. Hooper ([Hoo66]), is the basic undecidability result which all others will be derived from.

Theorem 3.2.2. *The mortality of* deterministic *Turing machines is undecidable.*

Since deterministic Turing machines are a special case of non-deterministic machines, one immediately gets

Corollary 3.2.3. *The mortality of (deterministic) Turing machines is undecidable.*

A related, though (as will be shown) weaker property rules out only case (iii) above.

Definition 3.2.4 (Boundedness)**.** *A Turing machine $\mathcal{M}$ is* bounded *iff a global bound $L_{\mathcal{M}}$ on the tape span of arbitrary computation fragments exist. In other words, iff for all IDs $C = \langle q, m, f \rangle$, $C' = \langle q', m', f' \rangle$ with $C \vdash^\star C'$ one has $|m' - m| \leq L_{\mathcal{M}}$.*

Boundedness can alternatively be defined by limiting the number of reachable IDs instead of the maximal distance of the head from its initial position. These two definitions can, however, be used interchangeably, due to

Lemma 3.2.5. *A Turing machine $\mathcal{M}$ is* bounded *iff there is a $\widetilde{L}_{\mathcal{M}}$ such that for any ID C no more than $\widetilde{L}_{\mathcal{M}}$ IDs are reachable from C.*

Proof. Let $C = \langle q, m, f \rangle$ be an arbitrary ID of a bounded machine $\mathcal{M}$. Then for all $C' = \langle q', m', f' \rangle$ with $C \vdash^\star C'$ one has $m' \in I := [m - L_{\mathcal{M}}, m + L_{\mathcal{M}}]$. Furthermore, $f'(n) \neq f(n)$ only for $n \in I$ since during every step, only the cell beneath the head can be overwritten. This limits the number of different such C' to at most $\widetilde{L}_{\mathcal{M}} := |\mathcal{Q}| \cdot |I| \cdot |\mathcal{A}|^{|I|}$.

For the converse, let $C = \langle q, m, f \rangle$ be an arbitrary ID of a machine $\mathcal{M}$ where only $\widetilde{L}_M$ IDs are reachable from any single initial ID. For all these reachable IDs $C' = \langle q', m', f' \rangle$ one then has $|m' - m| < \widetilde{L}_{\mathcal{M}} =: L_{\mathcal{M}}$ since the head only moves by one cell during every transition. ■

The rest of the section will establish that boundedness, like mortality, is undecidable. This will be done by showing that once a Turing machine's boundedness is settled, its mortality becomes decidable. Many of the proofs on the way towards that goal depend on the similarity of certain IDs in the sense that they allow, for a while, the same transitions to occur. This idea is captured by the following definition and its corollary.

Definition 3.2.6 (n-Equivalency)**.** *Two IDs $C_1 = \langle q_1, m_1, f_1 \rangle$, $C_2 = \langle q_2, m_2, f_2 \rangle$ are n-equivalent, written $C_1 \equiv_n C_2$, iff*

$$q_1 = q_2$$
$$f_1(m_1 + \Delta) = f_2(m_2 + \Delta) \quad \text{for } |\Delta| \leq n.$$

Two n-equivalent IDs are thus indistinguishable for computations up to a tape span of n. This is formalized by

Corollary 3.2.7. *Let $n \in \mathbb{N}$ be arbitrary.*

(i) Let $C_0^1 = \langle q_0, m^1, f_0^1 \rangle$, $C_0^2 = \langle q_0, m^2, f_0^2 \rangle$ be IDs with $C_0^1 \equiv_n C_0^2$. Assume that C_0^1 initiates a computation (fragment) $C_0^1 \vdash \cdots \vdash C_l^1 = \langle q_l, m^1 + \Delta_l, f_l^1 \rangle$ with $\operatorname{span}_i C_i^1 =: k \leq n$. Then C_0^2 also initiates a computation (fragment) $C_0^2 \vdash \cdots \vdash C_l^2 = \langle q_l, m^2 + \Delta_l, f_l^2 \rangle$ with $\operatorname{span}_i C_i^2 = k$ and $f_l^1(m_0^1 + \delta) = f_l^2(m_0^2 + \delta)$ for $|\delta| \leq n$.

(ii) For any set of IDs $\mathcal{C}$, regardless of its size, $\mathcal{C}/\equiv_n$ is finite.

Proof.

(i). For $1 \leq i \leq k$ let $C_i^2 := \langle q_i, m^2 + \Delta_i, f_i^2 \rangle$ with

$$f_i^2(m^2 + \delta) := \begin{cases} f_i^1(m^1 + \delta) & \text{for } |\delta| \leq n, \\ f_0^2(m^2 + \delta) & \text{otherwise.} \end{cases}$$

Now $\operatorname{span}_i C_i^1 \leq n$ implies $\Delta_i \leq n$ and hence $f_i^2(m^2 + \Delta_i) = f_i^1(m^2 + \Delta_i)$ for all $0 \leq i \leq l$. Thus $C_i^2 \vdash C_{i+1}^2$ follows from $C_i^1 \vdash C_{i+1}^1$. The other required properties of the C_i^2 are true by the definitions of C_i^2 and f_i^2.

(ii). There are only $|\mathcal{Q}| \cdot |\mathcal{A}|^{2n+1}$ non-n-equivalent IDs for every n. ■

This lemma is put to its first use to show that, contrary to the case of mortality, boundedness is not weakened by disregarding infinite IDs.

Lemma 3.2.8. *A Turing machine $\mathcal{M}$ is bounded iff it is bounded for all finite initial IDs.*

Proof. Let $L_{\mathcal{M}}$ be a bound for the tape spans of computation fragments initiated by finite IDs, and assume $C = \langle q, m, f \rangle$, $C_n = \langle q_n, m_n, f_n \rangle$ are infinite IDs with $C \vdash^n C_n$ and $\Delta := |m_n - m| > L_{\mathcal{M}}$. By replacing the contents of all cells with a distance greater than Δ from m with $\square$, one can obtain a finite ID $C' = \langle q', m', f' \rangle$ with $C' \equiv_\Delta C$. Corollary 3.2.7 then mandates the existence of another finite ID $C'_n = \langle q'_n, m'_n, f'_n \rangle$ for which $C' \vdash^n C'_n$ and $\left|m'_n - m'\right| = \Delta > L_{\mathcal{M}}$. This contradicts the bound $L_{\mathcal{M}}$ on the lengths of computation fragments initiated by finite IDs. ■

With these tools at one's disposal, one can show that mortality is decidable for bounded Turing machines. At its core, the following proof decides the question of mortality by exhaustively searching for looping computations within a finite subset of IDs known to adequately represent every possible ID of the machine. This finite subset's size and contents, however, depend on the *value* of $L_{\mathcal{M}}$ which is a slightly stronger prerequisite than such a bound's mere existence. The algorithm therefore contains a search for a suitable $L_{\mathcal{M}}$ interwoven into the search for looping computations, thus adding a modest amount of complexity to an otherwise trivial proof.

Lemma 3.2.9. *The mortality of bounded Turing machines is decidable.*

Proof. Let $\mathcal{C}$ be the set of all IDs of an arbitrary bounded Turing machine $\mathcal{M}$. For an arbitrary n, pick $C^1, \ldots, C^N$, one from every one of the N partitions of $\mathcal{C}$ modulo $\equiv_n$. Then compute the (finite!) sets of computation fragments of length j initiated by C^i,

$$\mathcal{C}_j^i := \{ C^i \vdash C_1 \ldots \vdash C_j \},$$

by first computing $\mathcal{C}_1^1, \ldots, \mathcal{C}_1^N$, then $\mathcal{C}_2^1, \ldots, \mathcal{C}_2^N$, and so on. Should one of the computations move the machine's head more than n cells away from its initial position then $n < L_{\mathcal{M}}$ and the procedure must be restarted with a larger n. Eventually though, n will exceed $L_{\mathcal{M}}$, and for such an n, after a while j will exceed the bound $\widetilde{L}_{\mathcal{M}}$ on the number of IDs reachable from a single initial ID (see 3.2.5). Any non-empty $\mathcal{C}_j^i$ must then contain a computation which reaches an ID twice, and which

thus loops. Hence, while computing these sets for ever increasing j, ultimately either all the sets $\mathcal{C}_j^i$ will be empty, proving the machine to be mortal, or one of them will contain a looping computation, proving the machine to be immortal. In both cases, the final value of n is a upper bound on the maximal distance of the head from its initial position. The generalization of $\mathcal{M}$'s behavior from the particular initial IDs $C^1,\ldots,C^N$ to an arbitrary initial ID is justified by 3.2.7. ■

Having dealt with the bounded case in 3.2.9, now the unbounded case needs to be tackled. For a deterministic machine and a single ID, it is clear that the computation initiated by that ID is bounded if it halts. But boundedness in general requires the bound on the distance the head moves from its initial position to be *independent* from a particular initial ID, and for non-deterministic machines also from the particular computation of the many that may originate from a single ID. This makes it conceivable that unboundedness does not in general imply immortality, since one could envision a sequence of computations, each halting eventually, but each taking the head further away from its initial position than its predecessor before doing so. As it turns out, though, from any such unbounded sequence of computations one can obtain an ID that initiates an infinite computation. The following theorem shows exactly this, albeit for a special case of such a sequence where each computation, in a way, extends its predecessor.

Lemma 3.2.10. *Let $(C_i)_{i\in\mathbb{N}}$ be a sequence of IDs with $C_i \equiv_i C_{i+1}$ where C_i initiates a computation fragment with a tape span of at least i. Then there is an ID C with $C \equiv_i C_i$ which initiates a computation with infinite tape span.*

Proof. First, note that if C_i is replaced by an i-equivalent ID with the head initially at cell zero, C_i still initiates a computation fragment whose tape span is at least i (due to 3.2.7), and $C_i \equiv_i C_{i+1}$ still holds. Since $C_i \equiv_i C_{i+1}$ also requires all the ID's to have the same initial state q, one can assume without loss of generality that

$$C_i = \langle q, 0, f_i \rangle.$$

In the case of equal initial head positions, from $C_i \equiv_i C_{i+1}$ is follows that $f_{i+1}|_{[-i,i]}$. Hence, for

$$C := \langle q, 0, f \rangle$$
$$f(i) := f_i(i),$$

one has $f|_{[-i,i]} = f_i|_{[-i,i]}$ and thus

$$C \equiv_i C_i.$$

Therefore, for every i, since C_i leads to a computation fragment with a tape span of at least i, by 3.2.7, C does too. Hence C initiates a computation with an infinite tape span. ■

The general case is handled by picking a suitable sub-sequence from an arbitrary unbounded sequence such that the requirements of 3.2.10 are fulfilled. This then finally proves

Lemma 3.2.11. *An unbounded Turing machine is necessarily immortal.*

Proof. Since the machine is unbounded, there exists an infinite sequence of IDs $(C_i)_{i\in\mathbb{N}}$ where C_i initiates a computation with a tape span of at least i. Since $C_i \equiv_i C_{i+1}$ will in general not hold for this sequence, one needs to pick a sub-sequence of $(C_i)_{i\in\mathbb{N}}$ such that it does. For that, remember that for every n, $\equiv_n$ splits a set of IDs into *finitely* many partitions (3.2.7). Any such partitioning of an infinite set will therefore contains at least one infinite partition. This guarantees the following to yield an infinitely decreasing series of non-empty subsets of $\{ C_i \mid i \in \mathbb{N} \}$.

$$\mathcal{C}_0 \in \{ C_i \mid i \in \mathbb{N} \}/\equiv_0 \text{ with } \mathcal{C}_0 \text{ infinite},$$
$$\mathcal{C}_{i+1} \in \{ C_j \mid C_j \in \mathcal{C}_i, j > i \}/\equiv_{i+1} \text{ with } \mathcal{C}_{i+1} \text{ infinite}.$$

Now, for any series $(\widetilde{C}_i)_{i\in\mathbb{N}}$ with

$$\widetilde{C}_i \in \mathcal{C}_i,$$

$\widetilde{C}_i \equiv_i \widetilde{C}_{i+1}$ and $\widetilde{C}_i$ leads to a computation whose tape span is at least i, both by the construction of $(\mathcal{C}_i)_{i\in\mathbb{N}}$. Thus, by 3.2.10, an ID C exists which initiates a computation with infinite tape span and hence also infinite length. ■

Although looking innocent at first sight, 3.2.11 is interesting enough to dwell on it for a moment before concluding this section with its main result.

3.2.11 can be restated, a bit provocatively, as "Any terminating algorithm has time and space complexity $O(1)$". Now that sounds rather contradictory – after all, most algorithms one deals with have no fixed upper bound on their execution time, yet one usually believes them to terminate regardless of their input. The question is, though, what kind of "input" one allows. 3.2.11 was derived from a definition of *mortality* that included *infinite* IDs. "Terminating" in the provocative restatement of that lemma thus means terminating even for all infinite inputs, while usually one expects to feed only a finite amount of information to an algorithm. 3.2.11 shows that this restriction is quite sensible, since most interesting algorithms (whose execution time depends on their input) cannot terminate for all infinite inputs.

Using the previous results about the relationship between mortality and boundedness, the main result of this section follows trivially

Theorem 3.2.12. *The boundedness of (deterministic) Turing machines is undecidable.*

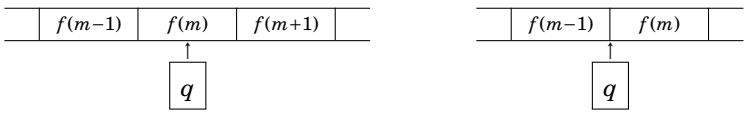
Proof. Since mortality is decidable for bounded machines (3.2.9) and unbounded machines are necessarily immortal (3.2.11), the decidability of boundedness implies that of mortality. The theorem thus follows from the undecidability of mortality (3.2.3). ■

3.3 Inter-Cell Turing Machines

The reachability relation $\vdash^\star$ of the Turing machines that were discussed so far is reflexive and transitive (by construction), but in general not symmetric. This stands in the way of the eventual goal of proving the boundedness problem of certain deduction system of term equalities to be undecidable by reducing the boundedness problem of Turning Machines to the former. To overcomes this issue, it it necessary to find a class of Turing machines for which $\vdash^\star$ *is* symmetric, and for which boundedness is still undecidable.

Unfortunately, for the Turing machines discussed so far, that approach doesn't seem feasible. One can transform a given Turing machine $\mathcal{M}$ into one with a symmetric transition relation easily enough by simply replacing $\mathcal{T}_\mathcal{M}$ by its symmetric closure, but the resulting symmetric machine has little in common with the original machine $\mathcal{M}$. The main issue is that the machine cannot always *read* the tape cell it wrote in the previous step, and so to allow it to undo the *particular* step it just performed, we need to allow it to perform *any* step backwards that would have brought it into the same state. This is remedied by first extending the results that were so far proven for regular Turning Machines to a class called *inter-cell Turing machines* which will then be shown to have a well-behaved subclass of symmetric machines.

An inter-cell Turing machine $\mathcal{M}$, like a regular Turing machine, is described by its finite alphabet $\mathcal{A}_\mathcal{M}$, again including the blank symbol $\square$, its finite state set $\mathcal{Q}_\mathcal{M}$ and a transition relation $\mathcal{T}_\mathcal{M} \subseteq \mathcal{Q}_\mathcal{M} \times \mathcal{A}_\mathcal{M} \times \mathcal{A}_\mathcal{M} \times \mathcal{Q}_\mathcal{M} \times \{-1, 1\}$. IDs are, again like those of regular Turing machines, described by triples $\langle q, m, f \rangle \in \mathcal{Q}_\mathcal{M} \times \mathbb{Z} \times (\mathbb{Z} \to \mathcal{A}_\mathcal{M})$. But while for regular Turing machines one pictures the head to be positioned above some cell m, an inter-cell Turing machine's head is assumed to be positioned *between* the $(m-1)$-th and the m-th cell. The following diagram illustrates the situation



Regular Machine Inter-Cell Machine

To proceed from one ID to a successor, an ITM can either move left, in which case it reads and overwrites the symbol to the left of the head, or move right, reading and overwriting the symbol to the right of the head. Just as for regular Turing machines, the pair of current state and symbol read defines the possible combinations of new state and symbol written as well as the direction of movement. The successor relation for ITMs is thus defined by

Definition 3.3.1 (Successor Relation for Inter-Cell Turing Machines)**.** *For an inter-cell Turing machine $\mathcal{M}$ and two IDs*

$$C = \langle q, m, f \rangle,$$
$$C' = \langle q', m + \Delta, f' \rangle$$

of $\mathcal{M}$, C' is a successor *of C, denoted by $C \vdash_\mathcal{M} C'$, iff*

$$\langle q, f(\widetilde{m}), f'(\widetilde{m}), q', \Delta \rangle \in \mathcal{T}_\mathcal{M},$$
$$f'(i) = f(i) \quad \text{for all } i \neq \widetilde{m}$$

with

$$\widetilde{m} = \begin{cases} m & \text{for } \Delta = +1 \\ m-1 & \text{for } \Delta = -1 \end{cases}$$

Some care has to be taken when defining *deterministic* ITMs. Assume for some state q and two different symbols a_1, a_2 that $\mathcal{T}_\mathcal{M}$ contains quintuples $\langle q, a_1, a_1, q, -1 \rangle$ and $\langle q, a_2, a_2, q, +1 \rangle$. For a regular TM, such a $\mathcal{T}_\mathcal{M}$ still qualifies as deterministic, since every combination of state and symbol uniquely determines a transition. For ITMs, however, which symbol (the left one or the right one) is read depends on the direction of movement. Should an ITM, while in state q, happen to find the symbol a_1 to the left of its head and a_2 to the right, $\mathcal{T}_\mathcal{M}$ would allow it to move both left *and* right – surely something that is undesirable for a deterministic ITM. To truly make an ITM

deterministic, an additional partial function $\mathcal{F}^{\Delta}_{\mathcal{M}} : \mathcal{Q} \to \{-1, +1\}$ where $\langle q, a, a', q', \Delta\rangle \in \mathcal{T}_{\mathcal{M}}$ iff $\mathcal{F}^{\Delta}_{\mathcal{M}}(q) = \Delta$ must therefore exist, in addition to it being deterministic as a regular TM. In other words, a deterministic ITM's direction of movement must only depend on its internal state, not on the tape's contents.

Any regular Turing machine can be transformed into an inter-cell Turing machine, and even into one whose alphabet contains only two symbols, as shown by

Lemma 3.3.2. *Let $\mathcal{M}$ be a regular Turing machine. Then there is an effective construction of an inter-cell Turing machine $\widetilde{\mathcal{M}}$ over the alphabet $\mathcal{A}_{\widetilde{\mathcal{M}}} := \{\square = 0, 1\}$ which is bounded (mortal, deterministic) iff $\mathcal{M}$ is bounded (mortal, deterministic).*

Proof. Let $\mathcal{A}_{\mathcal{M}}$ be the alphabet of $\mathcal{M}$ and $\mathcal{Q}_{\mathcal{M}}$ its set of states. Since $\widetilde{\mathcal{M}}$ is supposed to have the alphabet $\{0, 1\}$, the symbols from $\mathcal{A}_{\mathcal{M}}$ need to be encoded as words over $\{0, 1\}$. This is done by assigning numbers to the individual elements of $\mathcal{A}_{\mathcal{M}}$, starting with 0 for the blank symbol $\square$. Mapping each symbol to its binary representation then gives the desired encoding. From now on, thus assume

$$\mathcal{A}_{\mathcal{M}} = \{\square = a_0, a_1, \ldots, a_{|\mathcal{A}_{\mathcal{M}}|-1}\}, \qquad N \in \mathbb{N} \text{ such that } |\mathcal{A}_{\mathcal{M}}| \leq 2^N,$$

and to ease notation also

$$v_{n]} = v \bmod 2^n \text{ (v shortened to } n \text{ bits)}, \qquad v_{[n]} = \left\lfloor \frac{v_{n]}}{2^{n-1}} \right\rfloor \text{ (the } n\text{-th bit)}.$$

A particular tape of $\mathcal{M}$ shall then corresponds to a tape of $\widetilde{\mathcal{M}}$ with N cells for every single cell of $\mathcal{M}$'s tape, containing the the binary encoding of that cell's content. $\mathcal{M}$ having its head positioned over a certain cell shall corresponds to $\widetilde{\mathcal{M}}$'s head having the first bit of that cell's encoding to its right. Thus, an ID $C = \langle q, m, f\rangle$ of $\mathcal{M}$ corresponds to an ID $\widetilde{C} = \langle \widetilde{q}, \widetilde{m}, \widetilde{f}\rangle$ of $\widetilde{\mathcal{M}}$ iff for all $n \in \mathbb{Z}$

$$\widetilde{f}(\widetilde{m} + n) = v_{[n \bmod N]} \text{ with } a_v = f\left(m + \left\lfloor \frac{n}{N} \right\rfloor\right).$$

Note that whether an ID C of $\mathcal{M}$ that corresponds to an ID $\widetilde{C}$ of $\widetilde{\mathcal{M}}$ exists depends entirely on the existence of a $\widetilde{q}$ corresponding to q, and vice versa. Before delving into the precise definition of $\mathcal{Q}_{\widetilde{\mathcal{M}}}$ and its relation to $\mathcal{Q}_{\mathcal{M}}$, it seems beneficial to give a short overview of how $\widetilde{\mathcal{M}}$ simulates one step of $\mathcal{M}$.

Assume that $\widetilde{\mathcal{M}}$ starts from an ID that corresponds to some ID of $\mathcal{M}$. $\widetilde{\mathcal{M}}$ must then first read the N cells to the right of its head, since those represent the symbol under $\mathcal{M}$'s head. Having accumulated that symbol in its internal state, $\widetilde{\mathcal{M}}$ needs to consult $\mathcal{M}$'s transition relation to let its state reflect the new state $\mathcal{M}$ would be in, which symbol to write, and in which direction to move. Finally, it needs to carry out those actions, overwriting the previously read N cells with the new symbol and then moving by N cells to the left or right. $\widetilde{\mathcal{M}}$'s actions through one such cycle are controlled by the command component $\mathcal{C}_N$ of its internal state, which represents the current phase (read, write or move) and how many of the N cells visited during each of these phases have already been processed.

$$\mathcal{C}_N := \{R_1, \ldots, R_N, W_N, \ldots, W_1, M_1, \ldots, M_N\}.$$

During each transitions, the state's command component is replaced by its successor, starting anew with R_1 after completing the simulation of one step of $\mathcal{M}$ by reaching M_N. As expected, $\widetilde{\mathcal{M}}$ thus takes $3N$ steps to simulate one step of $\mathcal{M}$.

Apart of that command component, $\widetilde{\mathcal{M}}$'s state must of course also reflect the state of $\mathcal{M}$ as well as which symbol it just read or must write, and which direction to move in. The full state set of $\widetilde{\mathcal{M}}$ is thus

$$\mathcal{Q}_{\widetilde{\mathcal{M}}} := \mathcal{Q}_{\mathcal{M}} \times \{0 \ldots |\mathcal{A}_{\mathcal{M}}| - 1\} \times \mathcal{C}_N \times \{-1, 0, 1\}.$$

A state of $\mathcal{M}$ corresponds to a state of $\widetilde{\mathcal{M}}$ where it is at the beginning of a simulation cycle. Hence q corresponds to $\widetilde{q}$ iff

$$\widetilde{q} = \langle q, 0, R_1, 0\rangle.$$

The construction of $\widetilde{\mathcal{M}}$ from $\mathcal{M}$ is concluded by defining its transition relation, which can easily be verified to implement the behavior outlined above.

$$
\begin{aligned}
\mathcal{T}_{\widetilde{\mathcal{M}}} := \Big\{\ & \big\langle\langle q,\ 0,\ R_1,\ 0\rangle,\ v_{[1]},\ 0,\ \langle q,\ v_{1]},\ R_2,\ 0\rangle,\ +1\big\rangle, \\
& \big\langle\langle q,\ v_{n-1]},\ R_n,\ 0\rangle,\ v_{[n]},\ 0,\ \langle q,\ v_{n]},\ R_{n+1},\ 0\rangle,\ +1\big\rangle, \\
& \big\langle\langle q,\ v_{[N-1]},\ R_N,\ 0\rangle,\ v_{[N]},\ 0,\ \langle q',\ v',\ W_N,\ d'\rangle,\ +1\big\rangle, \qquad \circledast \\
& \\
& \big\langle\langle q',\ v'_{N]},\ W_N,\ d'\rangle,\ 0,\ v'_{[N]},\ \langle q',\ v'_{N-1]},\ W_{N-1},\ d'\rangle,\ -1\big\rangle, \\
& \big\langle\langle q',\ v'_{n]},\ W_n,\ d'\rangle,\ 0,\ v'_{[n]},\ \langle q',\ v'_{n-1]},\ W_{n-1},\ d'\rangle,\ -1\big\rangle, \\
& \big\langle\langle q',\ v'_{1]},\ W_1,\ d'\rangle,\ 0,\ v'_{[1]},\ \langle q',\ 0,\ M_1,\ d'\rangle,\ -1\big\rangle, \\
& \\
& \big\langle\langle q',\ 0,\ M_1,\ d'\rangle,\ b,\ b,\ \langle q',\ 0,\ M_1,\ d'\rangle,\ d'\big\rangle, \\
& \big\langle\langle q',\ 0,\ M_n,\ d'\rangle,\ b,\ b,\ \langle q',\ 0,\ M_{n+1},\ d'\rangle,\ d'\big\rangle, \\
& \big\langle\langle q',\ 0,\ M_N,\ d'\rangle,\ b,\ b,\ \langle q',\ 0,\ R_1,\ 0\rangle,\ d'\big\rangle \\
& \\
& \Big|\ \langle q, a_v, a_{v'}, q', d'\rangle \in \mathcal{T}_{\mathcal{M}},\quad b \in \{0,1\},\quad 1 < n < N \qquad\qquad \Big\}.
\end{aligned}
$$

Observe now that during an arbitrary computation, $\widetilde{\mathcal{M}}$ will assume a state $\langle q, 0, R_1, 0\rangle$ ($q \in \mathcal{Q}_{\mathcal{M}}$) exactly every $3N$ steps. Since these are exactly the states of $\widetilde{\mathcal{M}}$ which correspond to states of $\mathcal{M}$, every $3N$-th ID in a computation of $\widetilde{\mathcal{M}}$ corresponds to an ID of $\mathcal{M}$. Furthermore, if $\widetilde{\mathcal{M}}$ reaches ID $\widetilde{C}$ and then $3N$ steps later reaches $\widetilde{C}'$, $C \vdash C'$ holds for the corresponding IDs C, C' of $\mathcal{M}$. Therefore, from every computation of $\widetilde{\mathcal{M}}$ with length $n \geq 3N$ one can obtain a computation of $\mathcal{M}$ with length $\lfloor \frac{n}{3N} \rfloor - 1$. Conversely, a computation of $\mathcal{M}$ can be transformed to a computation of $\widetilde{\mathcal{M}}$ with $3N$ times as many steps as the original. Thus, $\widetilde{\mathcal{M}}$ is bounded iff $\mathcal{M}$ is bounded.

Also, careful checking of $\mathcal{T}_{\widetilde{\mathcal{M}}}$ shows that all IDs of $\widetilde{\mathcal{M}}$ except those with command state R_N (marked with $\circledast$) have a uniquely defined successor. For those with command state R_N, the successor is uniquely defined iff $\mathcal{M}$'s transition relation uniquely defines q', $a_{v'}$ and d' for q and a. $\widetilde{\mathcal{M}}$ is therefore deterministic iff $\mathcal{M}$ is deterministic. ■

Now, combining 3.2.12 and 3.3.2 immediately proves

Theorem 3.3.3. *The boundedness of (deterministic) inter-cell Turing machines over an alphabet with two elements is undecidable.*

3.4 Symmetric Inter-Cell Turing Machines

As promised when inter-cell Turing machines where introduced, it is now shown that this class of Turing machines has a well-behaved subclass of machines with a symmetric transition relation.

Definition 3.4.1 (Symmetric ITMs)**.**

(i) An inter-cell Turing machine $\mathcal{M}$ is symmetric *iff for every $\langle q, a, a', q', \Delta\rangle \in \mathcal{T}_\mathcal{M}$ also $\langle q', a', a, q, -\Delta\rangle \in \mathcal{T}_\mathcal{M}$.*

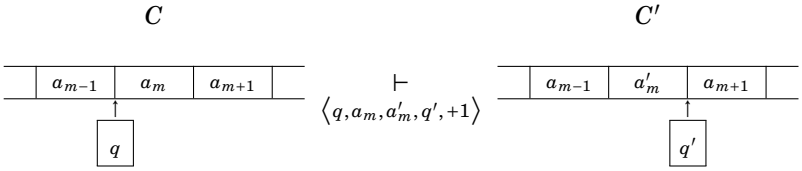
(ii) The symmetric closure *of an inter-cell Turing machine $\mathcal{M}$ is the symmetric inter-cell Turing machine $\overline{\mathcal{M}}$ which shares $\mathcal{M}$'s alphabet and states but whose transition relation is extended to*

$$\mathcal{T}_{\overline{\mathcal{M}}} := \{\, \langle q, a, a', q', \Delta\rangle, \langle q', a', a, q, -\Delta\rangle \mid \langle q, a, a', q', \Delta\rangle \in \mathcal{T}_\mathcal{M} \,\}$$

The following corollary asserts that this definition indeed fulfills its purpose. Note that while this definition of symmetricity applies to regular Turing machines also, this corollary does not!

Corollary 3.4.2. *An inter-cell Turing machine $\mathcal{M}$ is symmetric iff $\vdash^\star_\mathcal{M}$ is a symmetric relation.*

Proof. Since $\vdash^\star_\mathcal{M}$ is the reflexive and transitive closure of $\vdash_\mathcal{M}$, it is sufficient to prove the corollary for the latter. The following diagram shows a pair of IDs $C \vdash C'$ of an inter-cell Turing machine $\mathcal{M}$.



From the picture, it is clear that $C' \vdash C \Leftrightarrow C \vdash C'$ iff $\langle q, a, a', q', \Delta\rangle \in \mathcal{T}_\mathcal{M} \Leftrightarrow \langle q', a', a, q, -\Delta\rangle \in \mathcal{T}_\mathcal{M}$. This suffices, since $\vdash^\star$ if symmetric iff $\vdash$ is ($\vdash^\star$ is the reflexive and transitive closure of $\vdash$). ■

Assume now $\mathcal{M}$ is an arbitrary inter-cell Turing machine. Which properties of $\mathcal{M}$ do then carry over to its symmetric closure $\overline{\mathcal{M}}$ and which don't? Being deterministic clearly doesn't, since any computation fragment $C_1 \vdash_\mathcal{M} C_2 \vdash_\mathcal{M} C_3$, $C_1 \neq C_3$, of $\mathcal{M}$ mandates that on $\overline{\mathcal{M}}$ both $C_2 \vdash_{\overline{\mathcal{M}}} C_3$ and $C_2 \vdash_{\overline{\mathcal{M}}} C_1$ hold. $\overline{\mathcal{M}}$ is also generally immortal, since for any two IDs C, C' with $C \vdash_\mathcal{M} C'$ one has $C \vdash_{\overline{\mathcal{M}}} C' \vdash_{\overline{\mathcal{M}}} C \vdash \cdots$ on $\overline{\mathcal{M}}$. While not quite as obvious, $\overline{\mathcal{M}}$ will also in general be unbounded, as the following example shows.

Let $\mathcal{M}$ be an ITM over the one-element alphabet $\mathcal{A}_\mathcal{M} = \{\square\}$, with states $\mathcal{Q}_\mathcal{M} = \{q_2, q_1, q_0\}$ and with the transition relation

$$\mathcal{T}_\mathcal{M} := \left\{ \begin{array}{l} \langle q_2,\ \square,\ \square,\ q_0,\ +1\rangle, \\ \langle q_2,\ \square,\ \square,\ q_1,\ +1\rangle, \\ \langle q_1,\ \square,\ \square,\ q_0,\ +1\rangle \end{array} \right\}.$$

$\mathcal{M}$ is obviously bounded, since no computation with a length of 3 or greater exists. But the transition relation of $\mathcal{M}$'s symmetric closure $\overline{\mathcal{M}}$,

$$\mathcal{T}_{\overline{\mathcal{M}}} = \left\{ \begin{array}{l} \langle q_2,\ \square,\ \square,\ q_0,\ +1\rangle, \\ \langle q_0,\ \square,\ \square,\ q_2,\ -1\rangle, \\ \langle q_2,\ \square,\ \square,\ q_1,\ +1\rangle, \\ \langle q_1,\ \square,\ \square,\ q_2,\ -1\rangle, \\ \langle q_1,\ \square,\ \square,\ q_0,\ +1\rangle, \\ \langle q_0,\ \square,\ \square,\ q_1,\ -1\rangle \end{array} \right\}.$$

allows the computation

$$\langle q_2, 0, f\rangle \vdash_{\overline{\mathcal{M}}} \langle q_1, 1, f\rangle \vdash_{\overline{\mathcal{M}}} \langle q_0, 2, f\rangle \vdash_{\overline{\mathcal{M}}} \langle q_2, 1, f\rangle \vdash_{\overline{\mathcal{M}}} \cdots$$

to occur, which has an infinite tape span.

Taking a closer look at this computation of $\overline{\mathcal{M}}$ with infinite tape span reveals that its construction depends on $\overline{\mathcal{M}}$'s ability to move back to state q_0 via a different path than it was originally reached by. Otherwise, the head's movements would cancel out, and the computation would have an infinite length but a finite tape span. For originally deterministic $\mathcal{M}$, the symmetric closure lacks this ability, which is the essence of the proof of

Lemma 3.4.3. *A deterministic inter-cell Turing machines $\mathcal{M}$ is bounded iff its symmetric closure $\overline{\mathcal{M}}$ is bounded.*

Proof. Since any computation fragment on $\mathcal{M}$ is also a valid computation fragment on $\overline{\mathcal{M}}$, $\overline{\mathcal{M}}$ is obviously unbounded if $\mathcal{M}$ is. For the converse, let assume that $\mathcal{M}$ is bounded by $L_{\mathcal{M}}$ and that $\langle q_0, m_0, f_0\rangle = C_0 \vdash_{\overline{\mathcal{M}}} \cdots \vdash_{\overline{\mathcal{M}}} C_l = \langle q_l, m_l, f_l\rangle$ is a computation fragment with $|m_l - m_0| > 2 \cdot L_{\mathcal{M}}$ of $\overline{\mathcal{M}}$. By the definition of $\overline{\mathcal{M}}$'s transition relation, for all i one either has $C_i \vdash_{\mathcal{M}} C_{i+1}$ or $C_i \dashv_{\mathcal{M}} C_{i+1}$. Assume that there is an C_i such that $C_{i-1} \dashv_{\mathcal{M}} C_i \vdash_{\mathcal{M}} C_{i+1}$. Then $C_{i-1} = C_{i+1}$ since $\mathcal{M}$ is deterministic, and hence that sub-sequence can be replaced by just C_{i-1}. Note that this replacement changes neither the first nor the last ID of the computation fragment. Therefore, by performing such replacements until no suitable C_i is left, one eventually obtains a shortened computation fragment

$$C_0 \vdash_{\mathcal{M}} \cdots \vdash_{\mathcal{M}} C_r \dashv_{\mathcal{M}} \cdots \dashv C_l.$$

Since $2 \cdot L_M < |m_l - m_0| \leq |m_l - m_r| + |m_r - m_0|$, one of the computation fragments

$$C_0 \vdash_{\mathcal{M}} \cdots \vdash_{\mathcal{M}} C_r \text{ or}$$
$$C_l \vdash_{\mathcal{M}} \cdots \vdash_{\mathcal{M}} C_r$$

has a tape span larger than $L_{\mathcal{M}}$. ■

From this together with the undecidability of boundedness for deterministic inter-cell Turing machines (3.3.3) it immediately follows that

Theorem 3.4.4. *The boundedness of symmetric inter-cell Turing machines over an alphabet with two elements is undecidable.*

3.5 Strict Turing Machines and Strict IDs

The representation of a turing machine's tape as a function from $\mathbb{Z}$ to $\mathcal{A}$ was convenient for proving results about turing machines, but to reduce questions about turing machines to semi-unification the tape's contents must be interpreted as finite strings over some alphabet instead. Such an interpretation requires the very least a restriction to only *finite* IDs, but since these IDs still allow blank symbols to appear between non-blank ones, that class of IDs is still slightly too large. This is the reason for the introduction of *strict* IDs and *strict* Turing machines.

Definition 3.5.1 (Strict Turing Machines and Strict IDs)**.**

(i) A Turing machine is called strict *if* $\square \in \mathcal{A}_\mathcal{M}$*, but the blank symbol* $\square$ *does not occur in the transition relation. For strict Turing machines* $\mathcal{M}$*,* $\mathring{\mathcal{A}}_\mathcal{M}$ *denotes the non-blank symbols from* $\mathcal{A}$*, i.e.* $\mathring{\mathcal{A}}_\mathcal{M} = \mathcal{A}_\mathcal{M} \setminus \{\square\}$*, and it is generally assumed that* $\mathcal{Q}_\mathcal{M} \cap \mathcal{A}_\mathcal{M} = \emptyset$*.*

(ii) An ID $\langle q, m, f \rangle$ *is called* strict *if there are* $I_1, I_2 \in \mathbb{N}$ *such that*

$$f(i) \neq \square \quad \textit{iff } I_1 \leq i < I_2 \qquad \textit{and} \qquad I_1 \leq m \leq I_2$$

(iii) For a strict Turing machine, only strict IDs are considered unless specifically stated otherwise. In particular, boundedness *of a strict machine refers to boundedness for strict IDs, a* computation fragment *means a sequence of strict IDs, and so on.*

Since $\square$ does not occur in a strict machine's transition relation, such a machine cannot overwrite blank tape cells, nor replace non-blank cells with blank ones. As a consequence, a successor of a strict ID on a strict machine is again strict, with the *same* bounds I_1 and I_2. Such a machine can thus only reach *finitely* many IDs from each individual initial ID, which of course severely limits the computational expressiveness. Since the number of reachable IDs depends on the distance $I_2 - I_1$ between the bounds of an initial IDs, however, that does not imply that strict machines are always bounded! In fact, the boundedness problem for strict machines is equivalent to the boundedness problem for arbitrary machines, as the following theorem shows.

Theorem 3.5.2. *Assume that* $\mathcal{M}$ *is a ((symmetric) inter-cell) Turing machine. Let* $\mathcal{M}'$ *be the strict ((symmetric) inter-cell) Turing machine which arises from* $\mathcal{M}$ *by first replacing the original blank symbol* $\square$ *by the new symbol* $\widetilde{\square}$ *everywhere, and finally adding* $\square$ *back to the alphabet. Then,* $\mathcal{M}$ *is bounded iff* $\mathcal{M}'$ *is bounded.*

Proof. Let $\langle q_0, m_0, f_0 \rangle = C_0 \vdash \cdots \vdash C_n$ be a computation fragment of $\mathcal{M}$ with $\operatorname{span}_i C_i = N$. By setting $f'_i(m_0 + \Delta) = f_i(m_0 + \Delta)$ for $|n| \leq N$ and $f'_i(m_0 + \Delta) = \square$ otherwise, one obtains a sequence of strict IDs $C'_i = \langle q_i, m_i, f'_i \rangle$ of $\mathcal{M}'$ with $C'_0 \vdash \cdots \vdash C'_n$ and $\operatorname{span}_i C'_i = N$. Thus $\mathcal{M}'$ is bounded if $\mathcal{M}$ is bounded.

Conversely, let $\langle q_0, m_0, f'_0 \rangle = C'_0 \vdash \cdots \vdash C'_n$ be a computation fragment of $\mathcal{M}'$ containing only strict IDs with $\operatorname{span}_i C'_i = N$. Since $\mathcal{M}'$ stops upon reaching a blank cell, all cells inspected during that computation fragment must be non-blank. Thus, by setting $f_i(n) := f'_i(n)$ if $f'_i(n) \neq \square$ and $f_i(n) := a$ for some $a \in \mathcal{A}_\mathcal{M}$ otherwise, one obtains a computation fragment $\langle q_0, m_0, f_0 \rangle = C_0 \vdash \cdots \vdash C_n$ on $\mathcal{M}$ with the same tape span. $\mathcal{M}$ is hence bounded if $\mathcal{M}'$ is bounded. ■

Thus, since boundedness is in general undecidable for ((symmetric) inter-cell) Turing machines, the same follows for the boundedness problem of strict machines.

Theorem 3.5.3. *The boundedness of strict ((symmetric) inter-cell) Turing machines over an alphabet containing two non-blank symbols is undecidable.*

IDs of strict inter-cell Turing machines can readily be interpreted as finite strings consisting of the tape contents to the left of the head, followed by the state, and followed by the tape contents to the right of the head.

Definition 3.5.4 (ID Expressions)**.** *For strict inter-cell Turning Machines $\mathcal{M}$,*

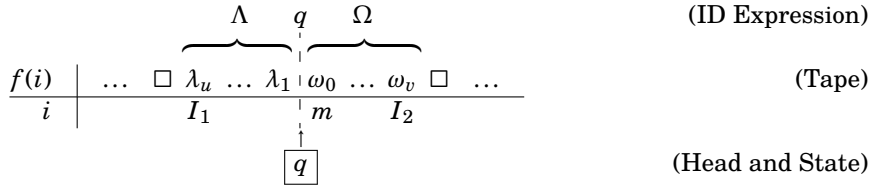
(i) An ID expression *is an element of the set*

$$\mathcal{I}_{\mathcal{M}} := \mathring{\mathcal{A}}^{\star}_{\mathcal{M}} \times \mathcal{Q}_{\mathcal{M}} \times \mathring{\mathcal{A}}^{\star}_{\mathcal{M}},$$

(ii) An ID expression $\Lambda q \Omega \in \mathcal{I}_{\mathcal{M}}$ corresponds *to an ID $\langle q, m, f \rangle$ of $\mathcal{M}$, denoted $\Lambda q \Omega \equiv \langle q, m, f \rangle$, iff for $\Lambda = \lambda_u \dots \lambda_1$, $\Omega = \omega_0 \dots \omega_v$*

$$f(m+\Delta) = \begin{cases} \lambda_{-\Delta} & \text{if } -u \le \Delta \le -1, \\ \omega_{\Delta} & \text{if } 0 \le \Delta \le v, \\ \square & \text{otherwise.} \end{cases}$$

The following diagram illustrates this correspondence. Note that since it was assumed that $\mathcal{A}_{\mathcal{M}}$ and $\mathcal{Q}_{\mathcal{M}}$ are disjoint, an ID expression $\Lambda q \Omega$ always uniquely specifies a head position within the tape contents $\Lambda\Omega$. The *absolute* head position m of an ID $\langle q, m, f \rangle$, on the other hand, is not reflected by an ID expression.



For strict *symmetric* inter-cell Turing machines (abbreviated *SSITM* from now on) $\mathcal{M}$, the transition relation $\mathcal{T}_{\mathcal{M}}$ can be interpreted as a set of *identities* between ID expressions of the form λq and those of the form $q\omega$. Each identity between two such ID expressions defines a replacement that can be applied to any matching ID expression $\Lambda q \Omega$, which moves the head exactly one cell to the left or to the right, and represents one computation step of the SSITM $\mathcal{M}$.

Definition 3.5.5. *For a strict symmetric inter-cell Turing machine (SSITM) $\mathcal{M}$,*

(i) The set of transition identities $\mathcal{T}^{\doteq}_{\mathcal{M}}$ *is*

$$\begin{aligned} \mathcal{T}^{\doteq}_{\mathcal{M}} := & \{\, q\omega \doteq \lambda q' \mid \langle q, \omega, \lambda, q', +1 \rangle \in \mathcal{T}_{\mathcal{M}} \,\} \\ & \cup \{\, \lambda q' \doteq q\omega \mid \langle q', \lambda, \omega, q, -1 \rangle \in \mathcal{T}_{\mathcal{M}} \,\} \end{aligned}$$

(ii) The successor relation $\vdash_{\mathcal{M}}$ *on $\mathcal{I}_{\mathcal{M}}$ is*

$$\Lambda q \Omega \vdash_{\mathcal{M}} \Lambda' q' \Omega' \quad \text{iff} \quad \Lambda q \Omega = \widetilde{\Lambda} \tau \widetilde{\Omega},\ \Lambda' q' \Omega' = \widetilde{\Lambda} \tau' \widetilde{\Omega}$$
$$\text{for some } \tau \doteq \tau' \in \mathcal{T}^{\doteq}_{\mathcal{M}} \text{ and } \widetilde{\Lambda}, \widetilde{\Omega} \in \mathring{\mathcal{A}}^{\star}_{\mathcal{M}}.$$

and $\vdash^{\star}_{\mathcal{M}}$ its transitive closure.

(iii) A sequence of ID expressions $\Lambda_0 q_0 \Omega_0 \vdash_{\mathcal{M}} \Lambda_1 q_1 \vdash_{\mathcal{M}} \cdots$ is called a computation fragment.

The following facts are immediate consequences of the definitions of $\mathcal{T}^{\doteq}_{\mathcal{M}}$ and $\vdash_{\mathcal{M}}$ and the fact that SSITMs cannot replace blank by non-blank symbols and vice versa.

Corollary 3.5.6. *For a SSITM $\mathcal{M}$,*

(i) $q\omega \doteq \lambda q' \in \mathcal{T}^{\doteq}_{\mathcal{M}}$ *iff* $\lambda q' \doteq q\omega \in \mathcal{T}^{\doteq}_{\mathcal{M}}$,

(ii) The relations $\vdash_{\mathcal{M}}$ and $\vdash^{\star}_{\mathcal{M}}$ on $\mathcal{I}_{\mathcal{M}}$ are symmetric,

(iii) If $\Lambda q \Omega \vdash^{\star}_{\mathcal{M}} \Lambda' q \Omega'$ then $|\Lambda\Omega| = |\Lambda'\Omega'|$.

Proof.

(i). By the definition of $\mathcal{T}^{\doteq}_{\mathcal{M}}$, $q\omega \doteq \lambda q' \in \mathcal{T}^{\doteq}_{\mathcal{M}}$ iff $\langle q, \omega, \lambda, q', +1 \rangle \in \mathcal{T}_{\mathcal{M}}$. Since $\mathcal{M}$ is symmetric, that is the case exactly if $\langle q', \lambda, \omega, q, -1 \rangle \in \mathcal{T}_{\mathcal{M}}$. But that holds, again by the definition of $\mathcal{T}^{\doteq}_{\mathcal{M}}$ exactly if $\lambda q' \doteq q\omega \in \mathcal{T}^{\doteq}_{\mathcal{M}}$.

(ii). Follows from (i).

(iii). Holds for $\vdash_{\mathcal{M}}$ by definition (cf. 3.5.5 (ii)), and thus also for $\vdash^{\star}_{\mathcal{M}}$. ∎

Calling a sequence $\Lambda_0 q_0 \Omega_0 \vdash_{\mathcal{M}} \Lambda_1 q_1 \vdash_{\mathcal{M}} \cdots$ of ID expressions a *computation fragment* is warranted by the following lemma. It shows that one can translate back and forth between "computation fragments" containing ID expressions, and computations containing IDs as per the original definition of that word.

Lemma 3.5.7. *Let $\mathcal{M}$ be an SSITM, then*

(i) For every computation fragment consisting of ID expressions

$$\Lambda_0 q_0 \Omega_0 \vdash_{\mathcal{M}} \Lambda_1 q_1 \Omega_1 \vdash_{\mathcal{M}} \cdots$$

there exists a corresponding computation fragment

$$\langle q_0, |\Lambda_0|, f_0\rangle \vdash_{\mathcal{M}} \langle q_1, |\Lambda_1|, f_1\rangle \vdash_{\mathcal{M}} \cdots \qquad \text{with } \Lambda_i q_i \Omega_i \equiv \langle q_i, |\Lambda_i|, f_i\rangle.$$

(ii) For every computation fragment

$$\langle q_0, m_0, f_0\rangle \vdash_{\mathcal{M}} \langle q_1, m_1, f_1\rangle \vdash_{\mathcal{M}} \cdots$$

there exists a corresponding computation fragment consisting of ID expressions

$$\Lambda_0 q_0 \Omega_0 \vdash_{\mathcal{M}} \Lambda_1 q_1 \Omega_1 \vdash_{\mathcal{M}} \cdots \qquad \text{with } \Lambda_i q_i \Omega_i \equiv \langle q_i, m_i, f_i\rangle \text{ where } m_i - |\Lambda_i| =: M \text{ is constant.}$$

Proof.

(i). Pick arbitrary IDs $\langle q_i, m_i, f_i\rangle$ with $\Lambda_i q_i \Omega_i \equiv \langle q_i, m_i, f_i\rangle$, and observe that these IDs can be shifted without interfering with the correspondence relation. Align the IDs such that their non-blank regions occupy the same *absolute* indices, and such that their (shared!) $I_1 = 1$ (where I_1 is as in definition 3.5.5). $m_i = |\Lambda_i|$ then follows from definition 3.5.4 (ii), and that the IDs are successors from a comparison of definitions 3.3.1 and 3.5.5.

(ii). Since SSITMs cannot replace blank by non-blank, nor non-blank by blank symbols, the I_1 and I_2 (as in definition 3.5.5) of all the IDs must agree. Let $M = m_0 - |\Lambda_0|$, then $|\Lambda_i| = m_i - m_0$ for all i follows from definition 3.5.4. That the ID expressions are successors again follows from a comparison of definitions 3.3.1 and 3.5.5. ■

The origin definition of the *tape span* of a computation fragment was stated in terms of absolute head positions, and hence does not directly apply to ID expressions. But it follows from lemma 3.5.7 that the position *relative* to the first (or equivalently to the last) non-blank symbol is a suitable replacement, which proves

Lemma 3.5.8. *For a computation fragment $C_0 \vdash_{\mathcal{M}} C_1 \vdash_{\mathcal{M}} \cdots$ and the corresponding computation fragment $\Lambda_0 q_0 \Omega_0 \vdash_{\mathcal{M}} \Lambda_1 q_1 \Omega_1 \vdash_{\mathcal{M}} \cdots$*

$$\operatorname*{span}_i C_i = \max_i \Big| |\Lambda_i| - |\Lambda_0| \Big| = \max_i \Big| |\Omega_i| - |\Omega_0| \Big|.$$

It follows that boundedness can also equivalently be defined in terms of the *relative* head position, and that it suffices to consider computation fragments where the head is positioned at the first non-blank cell in the beginning, and ends at the last non-blank cell.

Theorem 3.5.9. *For an SSITM $\mathcal{M}$ and some $L_{\mathcal{M}} \in \mathbb{N}$ the following propositions are equivalent*

(i) $\langle q, m, f\rangle \vdash^{\star}_{\mathcal{M}} \langle q', m', f'\rangle$ *implies* $|m' - m| \leq L_{\mathcal{M}}$*, i.e.* $\mathcal{M}$ *is* bounded *by bound* $L_{\mathcal{M}}$*,*

(ii) $\Lambda q \Omega \vdash^{\star}_{\mathcal{M}} \Lambda' q' \Omega'$ *implies* $\Big| |\Lambda'| - |\Lambda| \Big| = \Big| |\Omega'| - |\Omega| \Big| \leq L_{\mathcal{M}}$

(iii) $q\Omega \vdash^{\star}_{\mathcal{M}} \Lambda q'$ *implies* $|\Lambda| = |\Omega| \leq L_{\mathcal{M}}$.

Proof.

(i) ⇔ *(ii)*. This is an immediate consequence of lemma 3.5.8.

(ii) ⇒ *(iii)*. Setting $\Lambda = \Omega' = \varepsilon$ in (ii) yields (iv).

(iii) ⇒ *(ii)*. Assume (ii) fails, i.e. there is a computation fragment

$$\Lambda_0 q_0 \Omega_0 \vdash_{\mathcal{M}} \Lambda_1 q_1 \Omega_1 \vdash_{\mathcal{M}} \cdots \vdash_{\mathcal{M}} \Lambda_n q_n \Omega_n$$

with $\big||\Lambda_n| - |\Lambda_0|\big| > L_{\mathcal{M}}$. Let $a, b \in \{0, 1, \ldots, n\}$ such that $|\Lambda_a|$ and $|\Omega_b|$ are minimal. Then $|\Lambda_b| - |\Lambda_a| > L_{\mathcal{M}}$, Λ_a must be a common prefix of all the Λ_i (since the head needs to move past a symbol to modify it), and Ω_b must be a common suffix of all the Ω_i. Since $\Lambda_a q_a \Omega_a \vdash^\star_{\mathcal{M}} \Lambda_b q_b \Omega_b$, the same holds if these common parts are removed, which yields

$$q_a \Omega \vdash^\star_{\mathcal{M}} \Lambda q_b \quad \text{where } \Lambda_a \Lambda = \Lambda_b \text{ and } \Omega \Omega_b = \Omega_a.$$

Since $|\Lambda| = |\Lambda_b| - |\Lambda_a| > L_{\mathcal{M}}$, (iii) fails too. ■

Chapter 4

Decidability and Undecidability Results

4.1 The Undecidability of Semi-Unification

The stated goal of chapter 3 was to construct a class of Turing machines whose boundedness problem can be reduced to a boundedness problem for a certain system of path equations. The strict symmetric inter-cell Turing machines from section 3.5 allow exactly that. An ID of such a machine $\mathcal{M}$ can be represented by an expression of the form $\Lambda q \Omega$, where $\Lambda, \Omega \in \mathring{\mathcal{A}}_{\mathcal{M}} = \mathcal{A} \setminus \{\square\}$ are strings consisting of $\mathcal{M}$'s non-blank tape symbols, and $q \in \mathcal{Q}_{\mathcal{M}}$ is the machine's current state. These expressions are suspiciously similar to the path expressions introduced in 2.2, which had the form $\Pi \mu \Sigma$ where $\Pi \in \mathcal{P} = \{L, R\}^\star$ was a path, $\Sigma \in \mathcal{R} = \{S_1, S_2, \ldots\}^\star$, and $\mu \in \mathcal{V}$ a variable.

Thus, for a SSITM $\mathcal{M}$ whose tape alphabet contains only two non-blank symbols L, R and whose states are variables from $\mathcal{V}$, every ID expression of the form Λq is also a path expression, and vice versa. This correspondence is easily extended to *all* ID expressions and *all* path expressions by translating back and forth between $\Omega = \omega_{i_1} \ldots \omega_{i_n} \in \{L, R\}^*$ and $\Sigma = S_{i_1} \ldots S_{i_n} \in \mathcal{R}^\star$ by e.g. agreeing that L is represented by S_1 and R by S_2. This translation (in both directions) is the role of the map $\underline{\cdot}$.

This translation now allows the *transition identities* of an SSITM $\mathcal{M}$ with two non-blank symbols to be interpreted as a set of path equations $\Gamma_{\mathcal{M}}$. Note that the assumption $\mathring{\mathcal{A}}_{\mathcal{M}} = \{L, R\}$ and $\mathcal{Q}_{\mathcal{M}} \subset \mathcal{V}$ is no real restriction – if $\left|\mathring{\mathcal{A}}_{\mathcal{M}}\right| = 2$, one can always satisfy these assumptions by renaming the tape symbols and states.

Definition 4.1.1 (Path Equations for SSITMs)**.** *The set of path equations $\Gamma_{\mathcal{M}}$ corresponding to a SSITM $\mathcal{M}$ is*

$$\Gamma_{\mathcal{M}} := \left\{ \lambda q \doteq q' \underline{\omega} \;\middle|\; \lambda q \doteq q' \omega \in \mathcal{T}^{\doteq}_{\mathcal{M}} \right\},$$

assuming that $\mathring{\mathcal{A}}_{\mathcal{M}} = \{L, R\}$, $\mathcal{Q}_{\mathcal{M}} \subset \mathcal{V}$ and that $\underline{\cdot}$ is an isomorphism between $\mathring{\mathcal{A}}^\star_{\mathcal{M}}$ and $\{S_1, S_2\}^\star$. (Note that $\underline{\cdot}$ will be used to denote both the forward and the reverse translation).

The above took care of establishing a *formal* correspondence between the possible transitions of an SSITM $\mathcal{M}$ and a set of path equations $\Gamma_{\mathcal{M}}$. It remains to establish a *semantic* correspondence, i.e. a connection between the possible computation fragments on $\mathcal{M}$ and the path equations derivable from $\Gamma_{\mathcal{M}}$.

Without the additional bracketed premise of the deduction rule π-Application, it would hold that $\Pi \mu \Sigma \doteq \Pi' \nu \Sigma'$ being derivable from $\Gamma_{\mathcal{M}}$ is equivalent $\Pi \mu \underline{\Sigma}$ being reachable from $\Pi' \nu \underline{\Sigma'}$ and vice versa. *With* that additional premise of π-Application, this is no longer true – and it is thus necessary to find a property of computation fragments which reflects the additional restriction that the bracketed premise of π-Application represents.

That premise prevents one from "blindly" appending L or R to (the left of) a path expression, since doing so may produce, upon evaluation, undefined terms. On an SSITM, appending L or R to left of two mutually reachable ID expressions again yields two mutually reachable ID expressions – but there are two possible cases. It could be that appending the symbols allows some previously impossible computation fragment to occur, which takes actually takes the head to the very left of the extended ID expression at some point. Or it could be that the newly added symbol is "dead weight", and isn't actually used by *any* possible computation fragment. It turns out that the restricted version of π-Application used in definition 2.3.1 is equivalent to a restriction to the first case. Thus, derivable path equations correspond exactly to those mutually reachable IDs which don't carry any "dead weight" on their left hand side.

Theorem 4.1.2. *For an SSITM $\mathcal{M}$ with corresponding set of path equations $\Gamma_{\mathcal{M}}$,*

$$\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma}_{\mathcal{M}} \qquad \textit{iff} \qquad \Pi\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma''} \vdash^{\star}_{\mathcal{M}} \Pi'\nu\underline{\Sigma'} \quad \textit{for some } \sigma\Sigma'' \in \mathcal{I}_{\mathcal{M}}.$$

Proof of $\Rightarrow$. By induction on the structure of the derivation of a $\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \overline{\Gamma}_{\mathcal{M}}$.

$\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \in \Gamma_{\mathcal{M}}$. By the definition of $\Gamma_{\mathcal{M}}$, $\Sigma = \Pi' = \varepsilon$, and $\Pi\mu \doteq \nu\underline{\Sigma'} \in \mathcal{T}^{\doteq}_{\mathcal{M}}$. Thus, by the definition of $\vdash_{\mathcal{M}}$ for ID expressions, $\Pi\mu \vdash_{\mathcal{M}} \nu\underline{\Sigma'}$ and hence $\Pi\mu \vdash^{\star}_{\mathcal{M}} \nu\underline{\Sigma'} \vdash^{\star}_{\mathcal{M}} \nu\underline{\Sigma'}$.

$\frac{\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'}{\Pi'\nu\Sigma' \doteq \Pi\mu\Sigma}$ (Symmetricity). Per induction assumption, $\Pi\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma''} \vdash^{\star}_{\mathcal{M}} \Pi'\nu\underline{\Sigma'}$. Since $\vdash^{\star}_{\mathcal{M}}$ is symmetric, it follows that $\Pi'\nu\underline{\Sigma'} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma''} \vdash^{\star}_{\mathcal{M}} \Pi\mu\underline{\Sigma}$.

$\frac{\Pi_1\mu_1\Sigma_1 \doteq \Pi_2\mu_2\Sigma_2 \quad \Pi_2\mu_2\Sigma_2 \doteq \Pi_3\mu_3\Sigma_3}{\Pi_1\mu_1\Sigma_1 \doteq \Pi_3\mu_3\Sigma_3}$ (Transitivity). Per induction assumption, $\Pi_1\mu_1\underline{\Sigma_1} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \Pi_2\mu_2\underline{\Sigma_2}$ and $\Pi_2\mu_2\underline{\Sigma_2} \vdash^{\star}_{\mathcal{M}} \sigma'\underline{\Sigma'} \vdash^{\star}_{\mathcal{M}} \Pi_3\mu_3\underline{\Sigma_3}$. The transitivity of $\vdash^{\star}_{\mathcal{M}}$ then guarantees, as required, that $\Pi_1\mu_1\underline{\Sigma_1} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \Pi_3\mu_3\underline{\Sigma_3}$. (One could, of course, just as well use $\sigma'\underline{\Sigma'}$ instead of $\sigma\underline{\Sigma}$)

$\frac{\Pi\mu\Sigma \doteq \Pi'\nu\Sigma'}{\Pi\mu\Sigma S_i \doteq \Pi'\nu\Sigma' S_i} S_i$ (S_i-Application). Per induction assumption, $\Pi\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma''} \vdash^{\star}_{\mathcal{M}} \Pi'\nu\underline{\Sigma'}$. Since $\Box$ does not appear in $\mathcal{T}^{\doteq}_{\mathcal{M}}$, the same holds if S_i is appended everywhere, which yields, as required, $\Pi\mu\underline{\Sigma S_i} \vdash^{\star}_{\mathcal{M}} \sigma\underline{\Sigma''} \underline{S_i} \vdash^{\star}_{\mathcal{M}} \Pi'\nu\underline{\Sigma'}\underline{S_i}$.

$\frac{\Pi\mu\Sigma \doteq \Pi'\nu\Sigma' \quad [\widetilde{\Pi}\pi\Pi'\nu\Sigma' \doteq \ldots]}{\pi\Pi\mu\Sigma \doteq \pi\Pi'\nu\Sigma'} \pi$ (π-Application). Per induction assumption, $\Pi\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \Pi'\nu\underline{\Sigma'}$ and $\widetilde{\Pi}\pi\Pi'\nu\Sigma' \vdash^{\star}_{\mathcal{M}} \sigma\Sigma''$. Since the head can only move over one symbols at a time, $\widetilde{\Pi}\pi\Pi'\nu\Sigma' \vdash^{\star}_{\mathcal{M}} \sigma\Sigma''$ implies the existence of an ID expression $\sigma'\Sigma'''$ with $\widetilde{\Pi}\pi\Pi'\nu\Sigma' \vdash^{\star}_{\mathcal{M}} \widetilde{\Pi}\sigma'\Sigma'''$. For the *first* such ID expression, it must further hold that $\pi\Pi'\nu\Sigma' \vdash^{\star}_{\mathcal{M}} \sigma'\Sigma'''$, since none of the symbols in $\widetilde{\Pi}$ can have influenced the computation fragment up to $\sigma'\Sigma'''$. The proof is completed by appending π to $\Pi\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \Pi'\nu\underline{\Sigma'}$ – that, just as for S_i-Application produces ID expressions that are still mutually reachable – and together with the above it yields $\pi\Pi\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \sigma'\Sigma''' \vdash^{\star}_{\mathcal{M}} \pi\Pi'\nu\underline{\Sigma'}$.

Proof of $\Leftarrow$. It suffices to prove that if $\sigma\underline{\Sigma'} \vdash^{\star}_{\mathcal{M}} \Pi\mu\underline{\Sigma}$ then $\sigma\Sigma' \doteq \Pi\mu\Sigma \in \Gamma_{\mathcal{M}}$ – the general case then follows by suitable invocations of Symmetricity and Transitivity. The proof is carried out by induction on the length of the computation fragment $\sigma\underline{\Sigma'} \vdash^{\star}_{\mathcal{M}} \Pi\mu\underline{\Sigma}$.

$\sigma\underline{S_i\widetilde{\Sigma}} \vdash_{\mathcal{M}} \pi\mu\underline{\widetilde{\Sigma}}$. By the definitions of $\vdash_{\mathcal{M}}$ and $\mathcal{T}^{\doteq}_{\mathcal{M}}$ (and corollary 3.5.6), $\pi\mu \doteq \sigma S_i \in \mathcal{T}^{\doteq}_{\mathcal{M}}$, and hence $\pi\mu \doteq \mu S_i \in \Gamma_{\mathcal{M}}$. Using $\widetilde{\Sigma}$-Application and Symmetry, one can derive $\sigma S_i\widetilde{\Sigma} \doteq \pi\mu\widetilde{\Sigma}$ as required.

$\sigma\underline{\Sigma'} \vdash^{\star}_{\mathcal{M}} \widetilde{\Pi}\pi\nu\underline{\widetilde{\Sigma}} \vdash_{\mathcal{M}} \widetilde{\Pi}\mu\underline{S_i\widetilde{\Sigma}}$. The following derivation says it all

$$\dfrac{\sigma\Sigma' \doteq \widetilde{\Pi}\pi\nu\widetilde{\Sigma} \qquad \dfrac{\dfrac{\widetilde{\Pi}\pi\nu\underline{\widetilde{\Sigma}} \vdash_{\mathcal{M}} \widetilde{\Pi}\mu\underline{S_i\widetilde{\Sigma}}}{\mu S_i \doteq \pi\nu}}{\mu S_i\widetilde{\Sigma} \doteq \pi\nu\widetilde{\Sigma}}\,\widetilde{\Sigma} \qquad \dfrac{\text{(Induction Assumption)}}{\sigma\Sigma' \doteq \widetilde{\Pi}\pi\nu\widetilde{\Sigma} \in \overline{\Gamma}_{\mathcal{M}}}}{\widetilde{\Pi}\pi\nu\widetilde{\Sigma} \doteq \widetilde{\Pi}\mu S_i\widetilde{\Sigma}}\,\widetilde{\Pi}}{\sigma\Sigma' \doteq \widetilde{\Pi}\mu S_i\widetilde{\Sigma}}$$

■

With theorem 4.1.2, all the pieces for the promised reduction of the boundedness problem for SSITMs (over three element alphabets which include $\Box$) to the boundedness problem for finite systems of path equations are in place.

Theorem 4.1.3. *An SSITM $\mathcal{M}$ is bounded (as a strict Turing machine) iff $\Gamma_{\mathcal{M}}$ is bounded (as a set of path equations).*

Proof.

$\mathcal{M}$ is unbounded $\Rightarrow$ $\Gamma_{\mathcal{M}}$ is unbounded. If $\mathcal{M}$ is unbounded, per theorem 3.5.9 there exists a sequence of mutually reachable IDs $q_n\Omega_n \vdash^{\star}_{\mathcal{M}} \Lambda_n q'_n$ with $|\Lambda_n| \geq n$. According to theorem 4.1.2 that implies $q_n\underline{\Omega_n} \doteq \Lambda_n q'_n \in \overline{\Gamma}_{\mathcal{M}}$, and in particular $\Lambda_n q'_n \in \Delta_{\Gamma_{\mathcal{M}}}$, for all $n \in \mathbb{N}$. Since $|\Lambda| \geq n$, $\Gamma_{\mathcal{M}}$ is thus unbounded.

$\Gamma_{\mathcal{M}}$ is unbounded $\Rightarrow$ $\mathcal{M}$ is unbounded. If $\Gamma_{\mathcal{M}}$ is unbounded, there exists a sequence of path expressions $\Pi_n\mu\Sigma \in \Delta_{\Gamma}$ with $|\Pi_n| \geq n$. Thus, due to theorem 4.1.2, there are ID expressions $\sigma_n\Sigma'_n$ such that $\Pi_n\mu\underline{\Sigma} \vdash^{\star}_{\mathcal{M}} \sigma_n\Sigma'_n$. Since $|\Pi_n| \geq n$, that contradicts theorem 3.5.9 (ii), and $\mathcal{M}$ must therefore be unbounded. ■

From the equivalency of the boundedness of an SSITM $\mathcal{M}$ and the corresponding set of path equations $\Gamma_{\mathcal{M}}$, the undecidability of the latter follows immediately.

Theorem 4.1.4. *The boundedness and solvability of finite sets of path equations mentioning two local substitutions is, in general, undecidable.*

Proof. The boundedness of SSITMs over three-element alphabets (including $\Box$) is, according to theorem 3.5.3, in general undecidable. Such a boundedness problem can, via definition 4.1.1 and theorem 4.1.3, be *effectively* reduced to a boundedness problem for systems of path equations containing two local substitutions. The boundedness of such sets of path equations is thus, in general, undecidable as well. Since, according to 2.4.15, boundedness and solvability of a set of path equations is equivalent, the same holds for solvability of such sets. ■

Finally, by appealing to the reduction of *arbitrary* finite sets of path equations to semi-unification problems, the undecidability of semi-unification follows immediately.

Theorem 4.1.5. *The semi-unification problem for two inequalities is, in general, undecidable*

Proof. By theorem 2.2.21, the semi-unification problem for two inequalities is effectively equivalent to the solvability of finite sets of path equations mentioning two local substitutions, and this problem is in general undecidable per theorem 4.1.4. ■

4.2 The Decidability of Uniform Semi-Unification

The previous section showed that semi-unification problems involving at least two inequalities, or in other words more than two local substitutions, are in general undecidable. That leaves the question of decidability of semi-unification problems consisting of only a *single* inequality, called *uniform* semi-unification problems, open.

An analysis of the proof of theorem 4.1.5 quickly shows that at least *that* proof does not extend to the uniform case. Remember that, per definition 4.1.1, the tape contents to the right of the head are represented by S_1 or S_2, each representing one of the two non-blank tape symbols. Therefore, if the path equations $\Gamma_{\mathcal{M}}$ corresponding to a SSITM $\mathcal{M}$ are supposed to contain only a *single* local substitution S_1, the SSITM's alphabet must be restricted to at most *one* non-blank symbol. The question thus becomes whether theorem 3.5.3 about the undecidability of boundedness for SSITMs with two non-blank symbols holds for SSITMs with a single non-blank symbol as well. This, in turn, would require an extension of theorem 3.4.4 to the single-symbol case, i.e. that the boundedness of (not necessarily strict) symmetric inter-cell Turing machines is already undecidable for single-symbol alphabets. But that is clearly not the case – Turing machines over a single-symbol alphabet are simply finite state machines, and their boundedness is easily decided by looking for circles in their transition graph whose net head movement is non-zero.

Note, however, that this argument does not constitute a proof that uniform semi-unification is actually decidable! The encoding of Turing machines as sets of path equations in definition 4.1.1 is *not* surjective – since Turing machines cannot insert or delete symbols, only overwrite them, they only ever produce path equations $\Pi\mu\Sigma \doteq \Pi' \nu \Sigma'$ with $|\Pi| + |\Sigma| = \left|\Pi'\right| + \left|\Sigma'\right|$. One thus cannot deduce much about arbitrary uniform semi-unification problems by studying the behavior of single-element Turing machines.

Other methods are thus required to prove the decidability of uniform semi-unification. A very concise proof can be found in [Pud88].

Theorem 4.2.1 ([Pud88], page 552)**.** *The uniform semi-unification problem, i.e. the semi-unification problem for a single inequality, is decidable.*

The proof of Pudlák builds on the repeated-unification algorithm for semi-unification by Baaz, (cf. [Baa93]), and shows that for *uniform* semi-unification one always eventually either finds a semi-unifier, or recognizes the problem as unsolvable. That additional abortion condition in Pudlák's algorithm, (cf. [Pud88], page 553, (c)), can be viewed an extension of what is usually called the "occurs-check" in unification algorithms.

When a unification algorithm substitutes some term t for the variable α, the "occurs-check" tests whether t contains α. If that is the case, unification fails, since any such situation amounts to an attempt at unifying α and $f(\ldots, \alpha, \ldots)$, which is obviously impossible. Pudlák's additional termination condition serves the same purpose – to prevent an infinite loop if no solution exists – with its slightly higher complexity coming from the need to distinguish $\alpha \leq \alpha \rightharpoonup \alpha$, which *does* have a solution (e.g. $S = \mathrm{id}$, $S_1(\alpha) = \alpha \rightharpoonup \alpha$), from $\alpha \rightharpoonup \alpha \leq \alpha$, which does not.

In contrast, in the general case of more than one inequality, the undecidability result of the previous section shows that *no* exhaustive such check can exist.

Symbols

Index

Bibliography

[Baa93] M Baaz. “Note on the Existence of Most General Semi-Unifiers”. In: *Arithmetic, Proof Theory, and Computational Complexity*. Oxford University Press, May 1993, pp. 19–28. ISBN: 0198536909.

[BP93] M Baaz and P Pudlák. “Kreisel’s Conjecture for $L\exists_1$ ”. In: *Arithmetic, Proof Theory, and Computational Complexity*. Oxford University Press, May 1993, pp. 30–60. ISBN: 0198536909.

[DR92] Jochen Dörre and William C Rounds. “On subsumption and semiunification in feature algebras”. In: *Journal of Symbolic Computation* 13.4 (Apr. 1992), pp. 441–461. DOI: 10.1016/S0747-7171(08)80107-1.

[Hen88] Fritz Henglein. “Type inference and semi-unification”. In: *LFP ’88: Proceedings of the 1988 ACM conference on LISP and functional programming*. ACM, Jan. 1988. DOI: 10.1145/62678.62701.

[Hoo66] P K Hooper. “The undecidability of the Turing machine immortality problem”. In: *The Journal of Symbolic Logic* 31.02 (June 1966), pp. 219–234. DOI: 10.2307/2269811.

[JK93] S Jahama and A J Kfoury. “A general theory of semi-unification”. In: *Boston University Technical Reports* (1993). DOI: 10.1.1.1.9808.

[KTU88] A J Kfoury, J Tiuryn, and P Urzyczyn. “A proper extension of ML with an effective type-assignment”. In: *POPL ’88: Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, Jan. 1988, pp. 58–69. ISBN: 0897912527. DOI: 10.1145/73560.73565.

[KTU93] A J Kfoury, J Tiuryn, and P Urzyczyn. “The Undecidability of the Semi-unification Problem”. In: *Information and Computation* 102.1 (1993), pp. 83–101. DOI: 10.1006/inco.1993.1003.

[Pud88] P Pudlák. “On a unification problem related to Kreisel’s conjecture”. In: *Commentationes mathematicae Universitatis Carolinae* 29.3 (1988), pp. 551–556.