Forward Closure and the Finite Variant Property

Christopher Bouchard[1] Kimberly A. Gero[1]
Christopher Lynch[2] Paliath Narendran[1]

[1]University at Albany—SUNY, Albany, NY, USA

[2]Clarkson University, Potsdam, NY, USA

Frontiers of Combining Systems 2013

Motivation

Consider the following two theories:

- E_1: $f(g(u,x),\ g(v,y)) \approx g(f(u,v), f(x,y))$

- E_2: $f(f(x,y), f(x,y)) \approx f(x,y)$

Motivation

Consider the following two theories:

- E_1: $f(g(u,x),\, g(v,y)) \approx g(f(u,v),\, f(x,y))$
  - **Undecidable** unification problem*
- E_2: $f(f(x,y),\, f(x,y)) \approx f(x,y)$
  - **Decidable** unification problem
  - Can be shown by *forward closure*

* S. Anantharaman, et al. "Unification modulo Synchronous Distributivity." 2012.

Section 1

Introduction

Terms

- Purely syntactic
- Composed of. . .

Terms

- Purely syntactic
- Composed of...
  - Constants:

    $t_1 = a$

Terms

- Purely syntactic
- Composed of...

- Constants:

  $t_1 = a$

- Variables:

  $t_2 = x$

Terms

- Purely syntactic
- Composed of...

- Constants:

  $t_1 = a$

- Variables:

  $t_2 = x$

- Function Symbols:

  $t_3 = g(a, f(x))$

Terms

- Purely syntactic
- Composed of...

- Constants:

  $t_1 = a$

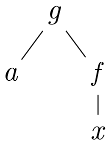
- Variables:

  $t_2 = x$

- Function Symbols:

  $t_3 = g(a, f(x))$

  $t_3 =$



Terms

- Purely syntactic
- Composed of...

- Constants:

  $t_1 = a$

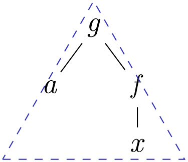
- Variables:

  $t_2 = x$

- Function Symbols:

  $t_3 = \quad g(a, f(x))$

  $t_3 =$



Terms

- Purely syntactic
- Composed of. . .

- Constants:

  $t_1 = a$

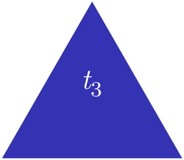
- Variables:

  $t_2 = x$

- Function Symbols:

  $t_3 = \quad g(a, f(x))$

  $t_3 =$



Rewriting

- Rewrite Rule: $t_1 \to t_2$
- Rewrite System: Set of rewrite rules

Example (Associativity)

$$f(x, f(y, z)) \quad \longrightarrow \quad f(f(x, y), z)$$

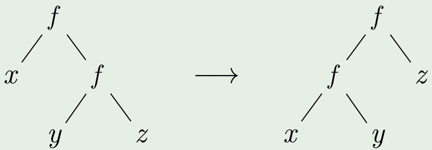
Rewriting

- Rewrite Rule: $t_1 \to t_2$

- Rewrite System: Set of rewrite rules

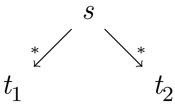
Example (Associativity)

$$f(x, f(y, z)) \quad \longrightarrow \quad f(f(x, y), z)$$



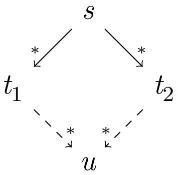
Convergence

- Confluence:



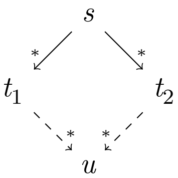
Convergence

- Confluence:



Convergence

- Confluence:

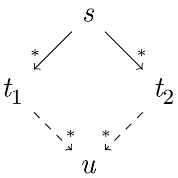


- Termination: No infinite descending chain

$$t_0 \to t_1 \to t_2 \to \cdots$$

Convergence

- Confluence:



- Termination: No infinite descending chain

$$t_0 \to t_1 \to t_2 \to \cdots$$

- Confluence + Termination = **Convergence**

Equational Unification

- Unification modulo a set of axioms E
- Given a set of equations $\mathcal{EQ} = \{\, s_1 \stackrel{?}{=} t_1,\, \ldots,\, s_n \stackrel{?}{=} t_n \,\}$
- Substitution σ is an E-unifier of $\mathcal{EQ}$ iff:

$$\sigma(s_1) \approx_E \sigma(t_1) \;\wedge\; \cdots \;\wedge\; \sigma(s_n) \approx_E \sigma(t_n)$$

Section 2

Motivation

Motivation

- Equational unification has lots of applications:
  - Automated reasoning
  - Programming languages (e.g., Maude)
  - Protocol analysis (e.g., Maude-NPA)

Motivation

- Equational unification has lots of applications:
  - Automated reasoning
  - Programming languages (e.g., Maude)
  - Protocol analysis (e.g., Maude-NPA)
- But equational unification is undecidable in general
  - Even when restricted to “nice” theories

Motivation

- How to identify decidable cases?

* C. Lynch and B. Morawska. "Basic Syntactic Mutation." 2002.

† H. Comon-Lundh and S. Delaune. "The finite variant property: How to get rid of some algebraic properties." 2005.

Motivation

- How to identify decidable cases?
- Two important syntactic approaches:
  - Basic Syntactic Mutation*
  - The Finite Variant Property†

* C. Lynch and B. Morawska. "Basic Syntactic Mutation." 2002.

† H. Comon-Lundh and S. Delaune. "The finite variant property: How to get rid of some algebraic properties." 2005.

Motivation

- How to identify decidable cases?
- Two important syntactic approaches:
  - Basic Syntactic Mutation*
  - The Finite Variant Property†
- Forward closure unifies these approaches.

* C. Lynch and B. Morawska. "Basic Syntactic Mutation." 2002.

† H. Comon-Lundh and S. Delaune. "The finite variant property: How to get rid of some algebraic properties." 2005.

Section 3

Forward Closure

Forward Closure

- Extension of work by Hermann on chain properties*
- Compress rewriting steps
- New rules capture chains of original rules

* M. Hermann. "Chain Properties of Rule Closures." 1990.

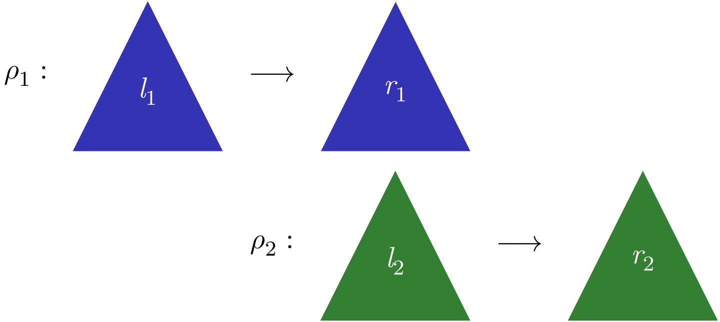
Overlap

- Overlap two rules and get a new rule
- General idea:

$$\text{If } t_1 \xrightarrow[\rho_1]{\epsilon} t_2 \xrightarrow[\rho_2]{p} t_3 \text{ then } t_1 \xrightarrow[\rho_1 \rightsquigarrow_p \rho_2]{\epsilon} t_3$$

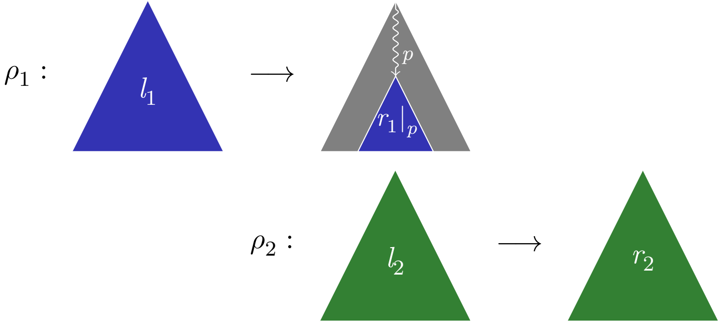
Overlap

- To compute $\rho_1 \rightsquigarrow_p \rho_2$ for $p \in \mathcal{FP}os(r_1)$:



Overlap

- To compute $\rho_1 \rightsquigarrow_p \rho_2$ for $p \in \mathcal{FP}os(r_1)$:

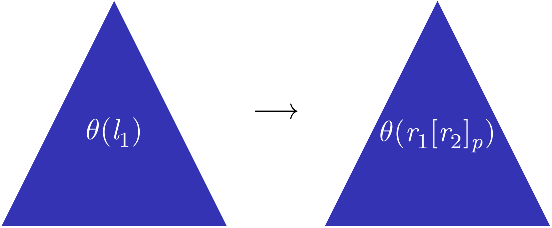


Overlap

$$\theta = \mathrm{mgu}(r_1|_p \stackrel{?}{=} l_2)$$

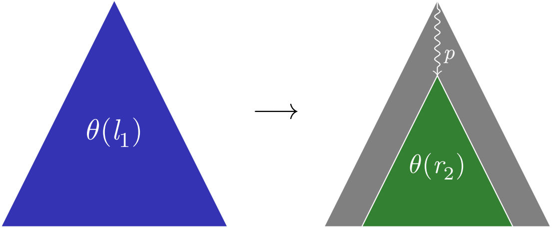
Overlap

$\rho_1 \rightsquigarrow_p \rho_2$:



Overlap

$\rho_1 \rightsquigarrow_p \rho_2$:



Overlap

Example

$$\rho_1 : \quad f(g(u_1, x_1),\ g(v_1, y_1)) \to g(f(u_1, v_1),\ f(x_1, y_1))$$
$$\rho_2 : \quad f(g(u_2, x_2),\ g(v_2, y_2)) \to g(f(u_2, v_2),\ f(x_2, y_2))$$

Overlap

Example

$$\rho_1: \quad f(g(u_1, x_1),\ g(v_1, y_1)) \to g(f(u_1, v_1),\ f(x_1, y_1))$$
$$\rho_2: \quad f(g(u_2, x_2),\ g(v_2, y_2)) \to g(f(u_2, v_2),\ f(x_2, y_2))$$

$\rho_1 \rightsquigarrow_1 \rho_2$:

Overlap

Example

$$\rho_1\colon \quad f(g(u_1, x_1),\ g(v_1, y_1)) \to g(f(u_1, v_1),\ f(x_1, y_1))$$
$$\rho_2\colon \quad f(g(u_2, x_2),\ g(v_2, y_2)) \to g(f(u_2, v_2),\ f(x_2, y_2))$$

$$\theta\colon \quad \{u_1 \mapsto g(u_2, x_2),\ v_1 \mapsto g(v_2, y_2)\}$$

$\rho_1 \rightsquigarrow_1 \rho_2\colon$

Overlap

Example

$$\rho_1 : \quad f(g(u_1, x_1),\ g(v_1, y_1)) \to g(f(u_1, v_1),\ f(x_1, y_1))$$
$$\rho_2 : \quad f(g(u_2, x_2),\ g(v_2, y_2)) \to g(f(u_2, v_2),\ f(x_2, y_2))$$

$$\theta : \quad \{u_1 \mapsto g(u_2, x_2),\ v_1 \mapsto g(v_2, y_2)\}$$

$$\rho_1 \rightsquigarrow_1 \rho_2 : \quad f(g(g(u_2, x_2), x_1),\ g(g(v_2, y_2), y_1)) \to g(g(f(u_2, v_2),\ f(x_2, y_2)),\ f(x_1, y_1))$$

Overlap

Example

$$\rho_1: \quad f(g(u_1, x_1),\ g(v_1, y_1)) \to g(f(u_1, v_1),\ f(x_1, y_1))$$
$$\rho_2: \quad f(g(u_2, x_2),\ g(v_2, y_2)) \to g(f(u_2, v_2),\ f(x_2, y_2))$$

$$\theta: \quad \{u_1 \mapsto g(u_2, x_2),\ v_1 \mapsto g(v_2, y_2)\}$$

$$\rho_1 \rightsquigarrow_1 \rho_2: \quad f(g(g(u_2, x_2), x_1),\ g(g(v_2, y_2), y_1)) \to g(g(f(u_2, v_2),\ f(x_2, y_2)),\ f(x_1, y_1))$$

Overlap

Example

$$\rho_1: \quad f(g(u_1, x_1),\ g(v_1, y_1)) \to g(f(u_1, v_1),\ f(x_1, y_1))$$
$$\rho_2: \quad f(g(u_2, x_2),\ g(v_2, y_2)) \to g(f(u_2, v_2),\ f(x_2, y_2))$$

$$\theta: \quad \{u_1 \mapsto g(u_2, x_2),\ v_1 \mapsto g(v_2, y_2)\}$$

$$\rho_1 \rightsquigarrow_1 \rho_2: \quad f(g(g(u_2, x_2), x_1),\ g(g(v_2, y_2), y_1))$$
$$\to g(g(f(u_2, v_2),\ f(x_2, y_2)),\ f(x_1, y_1))$$

Redundancy

- Not all overlaps are added to the forward closure.
- A rule $l \to r$ is *redundant* in a rewrite system R iff:

Redundancy

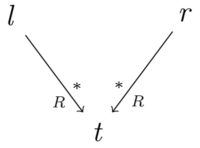
- Not all overlaps are added to the forward closure.
- A rule $l \to r$ is *redundant* in a rewrite system R iff:
  - $l \to r$ is an instance of a more general rule in R, or

Redundancy

- Not all overlaps are added to the forward closure.
- A rule $l \to r$ is *redundant* in a rewrite system R iff:
  - $l \to r$ is an instance of a more general rule in R, or
  - Every ground instance of $l \approx r$ can be proven by smaller ground instances of rules in R

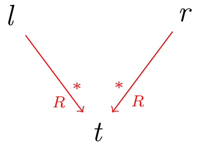
Redundancy

- Not all overlaps are added to the forward closure.
- A rule $l \to r$ is *redundant* in a rewrite system R iff:
  - $l \to r$ is an instance of a more general rule in R, or
  - Every ground instance of $l \approx r$ can be proven by smaller ground instances of rules in R



Redundancy

- Not all overlaps are added to the forward closure.
- A rule $l \to r$ is *redundant* in a rewrite system R iff:
  - $l \to r$ is an instance of a more general rule in R, or
  - Every ground instance of $l \approx r$ can be proven by smaller ground instances of rules in R



Computing Forward Closure

- Start with a rewrite system R

Computing Forward Closure

- Start with a rewrite system R
- Overlap each rule in R with each rule in R

Computing Forward Closure

- Start with a rewrite system R
- Overlap each rule in R with each rule in R
- Throw out redundant rules

Computing Forward Closure

- Start with a rewrite system R
- Overlap each rule in R with each rule in R
- Throw out redundant rules
- Call this $FC_1(R)$

Computing Forward Closure

- Start with a rewrite system $FC_k(R)$
- Overlap each rule in $FC_k(R)$ with each rule in R
- Throw out redundant rules
- Call this $FC_{k+1}(R)$

Computing Forward Closure

- Finally, $FC(R) = \bigcup_{k \geq 0} FC_k(R)$

Computing Forward Closure

- Finally, $FC(R) = \bigcup_{k \geq 0} FC_k(R)$
- If $FC_k(R) = FC_{k+1}(R)$ for some k, $FC(R)$ is finite
- Otherwise, $FC(R)$ is infinite

Forward Closure

- A term t is an *innermost redex* of R if it can only be rewritten by R at the root.

- Key Idea: In $FC(R)$, every innermost redex of R can be rewritten to its normal form in one step.

Section 4

Equivalence of FC and the FVP

Forward Closure and the Finite Variant Property

We show that a system has a finite forward closure if and only if it has the *finite variant property*.

Finite Variant Property

- The *variants* of t are all pairs (t', θ) such that:
  - θ is a normalized substitution
  - $\theta(t) \to^! t'$
- Variants capture the idea of rewriting to normal form
- Finite Variant Property: Every term has a finite set of most general variants

Equivalence of FC and the FVP

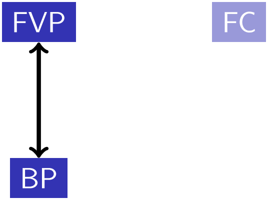
Boundedness Property

- Bound on the lengths of rewrite chains

- For each term t there is a bound $\#(t)$ such that

$$(\theta\downarrow)(t) \xrightarrow{\leq \#(t)} \theta(t)\downarrow$$

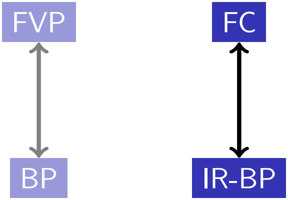
Equivalence of FC and the FVP



IR-Boundedness Property

- Bound on the lengths of rewrite chains from the root
- There is a *global* bound n such that, if a term t is an innermost redex, then $t \xrightarrow{\leq n} t{\downarrow}$.

Equivalence of FC and the FVP



Boundedness ⇒ IR-Boundedness

$$t \longrightarrow t{\downarrow}$$

- Suppose t is an innermost redex

Boundedness $\Rightarrow$ IR-Boundedness

$$f(t_1, \ldots, t_n) \longrightarrow t{\downarrow}$$

- Suppose t is an innermost redex
- Then $t = f(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are normalized

Boundedness $\Rightarrow$ IR-Boundedness

$$\theta(f(x_1, \ldots, x_n)) \longrightarrow t{\downarrow}$$

- Suppose t is an innermost redex
- Then $t = \theta(f(x_1, \ldots, x_n))$
- Where $\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is normalized

Boundedness ⇒ IR-Boundedness

$$\theta(t_f) \longrightarrow t\!\downarrow$$

- Suppose t is an innermost redex
- Then $t = \theta(t_f)$
- Where $\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is normalized
- And $t_f = f(x_1, \ldots, x_n)$

Boundedness ⇒ IR-Boundedness

$$\theta(t_f) \xrightarrow{\leq \#(t_f)} t{\downarrow}$$

- Suppose t is an innermost redex
- Then $t = \theta(t_f)$
- Where $\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is normalized
- And $t_f = f(x_1, \ldots, x_n)$

Boundedness ⇒ IR-Boundedness

$$\theta(t_f) \xrightarrow{\leq n} t{\downarrow}$$

- Suppose t is an innermost redex
- Then $t = \theta(t_f)$
- Where $\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is normalized
- And $t_f = f(x_1, \ldots, x_n)$
- Let $n = \max\{\#(t_f) \mid f \in \Sigma\}$

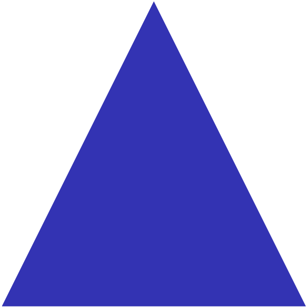
Boundedness $\Rightarrow$ IR-Boundedness

$$t \xrightarrow{\leq n} t{\downarrow}$$

- Suppose t is an innermost redex
- Then $t = \theta(t_f)$
- Where $\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is normalized
- And $t_f = f(x_1, \ldots, x_n)$
- Let $n = \max\{\#(t_f) \mid f \in \Sigma\}$

IR-Boundedness ⇒ Boundedness

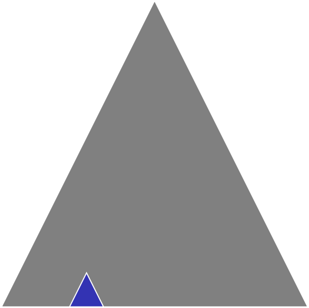
$$t \xrightarrow{\leq \#(t)} t{\downarrow}$$



$$\#(t) =$$

IR-Boundedness $\Rightarrow$ Boundedness

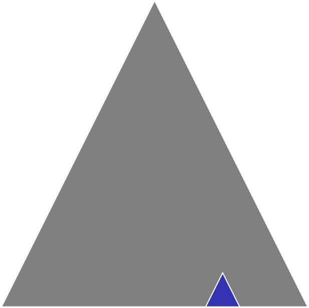
$$t \xrightarrow{\leq \#(t)} t\!\downarrow$$



$$\#(t) = n$$

IR-Boundedness $\Rightarrow$ Boundedness

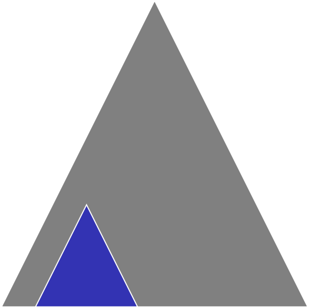
$$t \xrightarrow{\leq \#(t)} t{\downarrow}$$



$$\#(t) = n + n$$

IR-Boundedness $\Rightarrow$ Boundedness

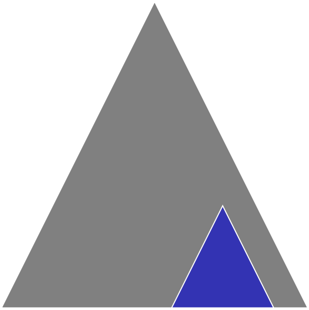
$$t \xrightarrow{\leq \#(t)} t{\downarrow}$$



$$\#(t) = n + n + n$$

IR-Boundedness $\Rightarrow$ Boundedness

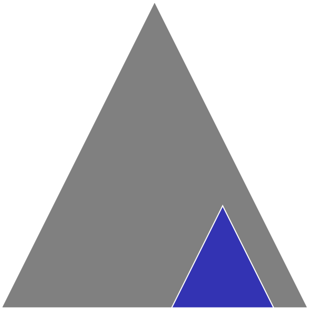
$$t \xrightarrow{\leq \#(t)} t{\downarrow}$$



$$\#(t) = n + n + n + n$$

IR-Boundedness $\Rightarrow$ Boundedness

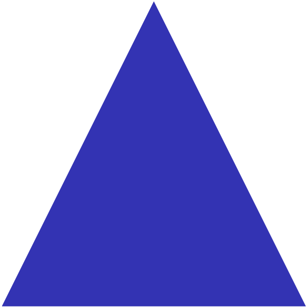
$$t \xrightarrow{\leq \#(t)} t{\downarrow}$$



$$\#(t) = n + n + n + n + \cdots$$

IR-Boundedness $\Rightarrow$ Boundedness

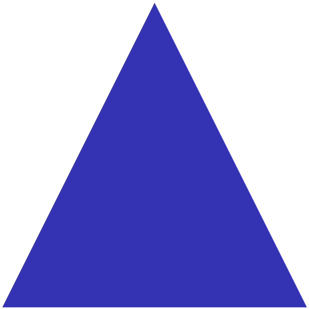
$$t \xrightarrow{\leq \#(t)} t\!\downarrow$$



$$\#(t) = n + n + n + n + \cdots \\ + n$$

IR-Boundedness ⇒ Boundedness

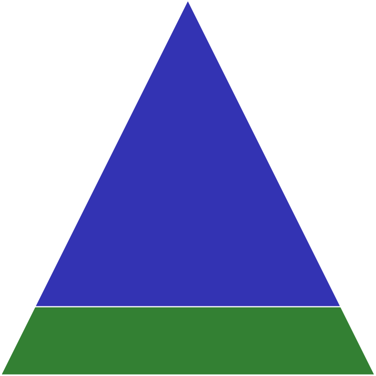
$$t \xrightarrow{\leq \#(t)} t{\downarrow}$$



$$\#(t) = n \cdot |\mathcal{FP}os(t)|$$

IR-Boundedness ⇒ Boundedness

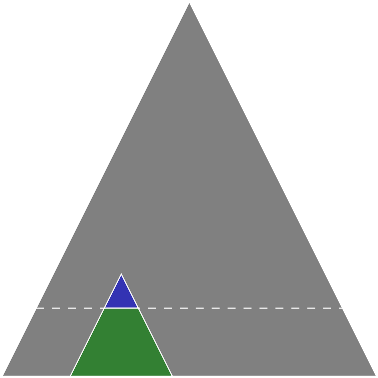
$$(\theta\downarrow)(t) \xrightarrow{\leq \#(t)} \theta(t)\downarrow$$



$$\#(t) = n \cdot |\mathcal{F}\mathcal{P}os(t)|$$

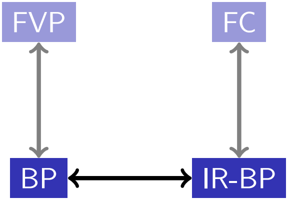
IR-Boundedness $\Rightarrow$ Boundedness

$$(\theta\!\downarrow)(t) \xrightarrow{\leq \#(t)} \theta(t)\!\downarrow$$

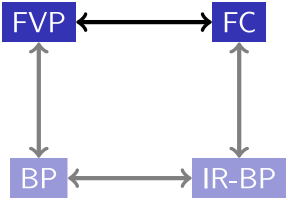


$$\#(t) = n \cdot |\mathcal{FP}os(t)|$$

Equivalence of FC and the FVP



Equivalence of FC and the FVP



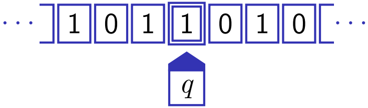
Section 5

Undecidability of FC

Undecidability of Finiteness of Forward Closure

- Reduction from the Uniform Mortality Problem for deterministic Turing machines

Uniform Mortality Problem



- Given a deterministic Turing machine
- Does every configuration halt in k or fewer steps (for some k)?
- Undecidable*

* G.G. Hillebrand, et al. "Undecidable Boundedness Problems for Datalog Programs." 1995.

Undecidability of Finiteness of Forward Closure

- Reduction from the Uniform Mortality Problem for deterministic Turing machines
- Start with a Turing machine M
- Create a rewrite system R_M
- $FC(R_M)$ is finite iff M is uniformly mortal

Section 6

Modularity of FC

Modularity

- Given rewrite systems R_1 and R_2, when does the following condition hold?

$$|FC(R_1)| + |FC(R_2)| < \infty \quad \Longrightarrow \quad |FC(R_1 \cup R_2)| < \infty$$

- We consider conditions on the signatures of R_1 and R_2.

Modularity for Disjoint Systems

- If R_1 and R_2 are rewrite systems with disjoint signatures
- Then $FC(R_1) \cup FC(R_2) = FC(R_1 \cup R_2)$.

Shared Constants

- If R_1 and R_2 share constants
- Then $FC(R_1 \cup R_2)$ may be infinite even if $FC(R_1)$ and $FC(R_2)$ are finite.

Example

$$R_1 := \{f(a, h(x)) \to h(f(b, x))\} \qquad R_2 := \{b \to a\}$$

$$f(a, h(x)) \to h(f(a, x)) \in FC(R_1 \cup R_2)$$

Shared Constants

- If R_1 and R_2 share constants
- Then $FC(R_1 \cup R_2)$ may be infinite even if $FC(R_1)$ and $FC(R_2)$ are finite.

Example

$$R_1 := \{f(a, h(x)) \to h(f(b, x))\} \qquad R_2 := \{b \to a\}$$

$$f(a, h(h(x))) \to h(h(f(b, x))) \in FC(R_1 \cup R_2)$$

Shared Constants

- If R_1 and R_2 share constants
- Then $FC(R_1 \cup R_2)$ may be infinite even if $FC(R_1)$ and $FC(R_2)$ are finite.

Example

$$R_1 := \{f(a, h(x)) \to h(f(b, x))\} \qquad R_2 := \{b \to a\}$$

$$f(a, h(h(x))) \to h(h(f(a, x))) \in FC(R_1 \cup R_2)$$

Summary of Results

- Finiteness of forward closure is equivalent to the finite variant property
- Finiteness of forward closure is undecidable
- Having the finite variant property is undecidable
- Finiteness of forward closure is preserved by union if the signatures are disjoint, but not if they share constants.

Future Work

- Forward closure modulo theory
- More detailed modularity results

Thank You

- cbou@cs.albany.edu

- Research supported in part by NSF grant CNS-0905286