On the Limits of Second-Order Unification

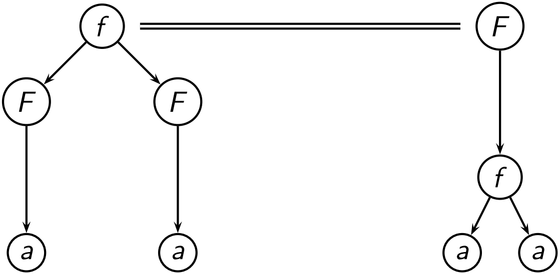
Jordi Levy
IIIA, CSIC, Barcelona, Spain

Joint work with Mateu Villaret,
Margus Veanes,
Manfred Schmidt-Schauss,
Temur Kutsia,...

UNIF'14, Viena

Second-Order Unification

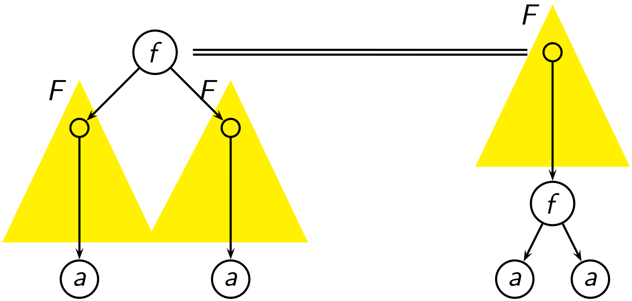
Variables may have arguments: $f(F(a), F(a)) \stackrel{?}{=} F(f(a,a))$



Instances of variables may use their arguments...

Second-Order Unification

Variables may have arguments: $f(F(a), F(a)) \stackrel{?}{=} F(f(a, a))$

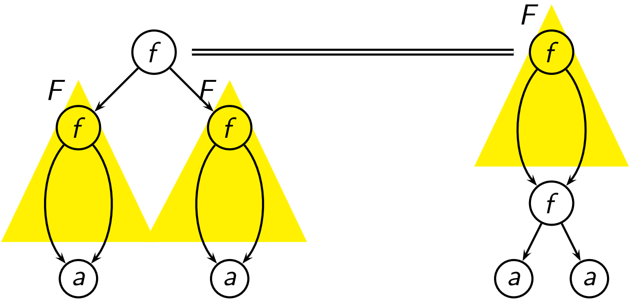


Instances of variables may use their arguments...

...just once like in $F \mapsto \lambda x \,.\, x$

Second-Order Unification

Variables may have arguments: $f(F(a), F(a)) \stackrel{?}{=} F(f(a,a))$



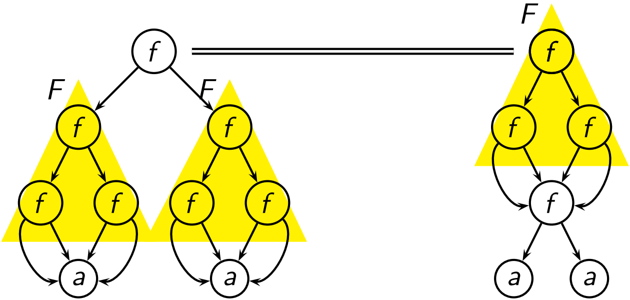
Instances of variables may use their arguments. . .

. . . just once like in $F \mapsto \lambda x \,.\, x$

. . . twice like in $F \mapsto \lambda x \,.\, f(x,x)$

Second-Order Unification

Variables may have arguments: $f(F(a), F(a)) \stackrel{?}{=} F(f(a,a))$



Instances of variables may use their arguments...

...just once like in $F \mapsto \lambda x\,.\,x$

...twice like in $F \mapsto \lambda x\,.\,f(x,x)$

...or more times like in $F \mapsto \lambda x\,.\,f(f(x,x), f(x,x))$

Variants of Second-Order Unification

Depending on the number of times instances may use variables we have

- Unrestricted: General **Second-Order Unification (SOU)**
- Just once: **Linear Second-Order Unification (LSOU)**
  Or **Context Unification (CU)** when variables may have at most one argument
- At most once: **Bounded Second-Order Unification (BSOU)**

All variants are **infinitary**:

$$F(f(a)) \stackrel{?}{=} f(F(a))$$

$$\{[F \mapsto \lambda x \,.\, f(.\overset{n}{.}.\, f(x) \dots)]\}_{n \geq 0}$$

Variants of Second-Order Unification

Depending on the number of times instances may use variables we have

- Unrestricted: General **Second-Order Unification (SOU)**
- Just once: **Linear Second-Order Unification (LSOU)**
  Or **Context Unification (CU)** when variables may have at most one argument
- At most once: **Bounded Second-Order Unification (BSOU)**

All variants are **infinitary**:

$$F(f(a)) \stackrel{?}{=} f(F(a))$$

$$\{[F \mapsto \lambda x \,.\, f(.\overset{n}{.}.\, f(x) \dots)]\}_{n \geq 0}$$

Variants of Second-Order Unification

Depending on the number of times instances may use variables we have

- Unrestricted: General **Second-Order Unification (SOU)**
- Just once: **Linear Second-Order Unification (LSOU)**
  Or **Context Unification (CU)** when variables may have at most one argument
- At most once: **Bounded Second-Order Unification (BSOU)**

All variants are **infinitary**:

$$F(f(a)) \stackrel{?}{=} f(F(a))$$

$$\{[F \mapsto \lambda x \,.\, f(.\overset{n}{.}.\, f(x) \ldots)]\}_{n \geq 0}$$

Variants of Second-Order Unification

Depending on the number of times instances may use variables we have

- Unrestricted: General **Second-Order Unification (SOU)**
- Just once: **Linear Second-Order Unification (LSOU)**
  Or **Context Unification (CU)** when variables may have at most one argument
- At most once: **Bounded Second-Order Unification (BSOU)**

All variants are **infinitary**:

$$F(f(a)) \stackrel{?}{=} f(F(a))$$

$$\{[F \mapsto \lambda x \,.\, f(.\overset{n}{.}.\, f(x)\dots)]\}_{n\geq 0}$$

General Un/Decidability Results

General SOU:

- **[Gould 1966]** Decidability of SO Matching
- **[Lucchesi 1972 and Huet 1973]** Third-Order Unification is undecidable
- **[Pietrzykowski 1973]** Complete SOU procedure
- **[Jensen and Pietrzykowski 1976]** Complete HOU procedure
- **[Goldfarb 1981]** SOU is undecidable
- **[Farmer 1988]** SOU is decidable if all function symbols are unary (Monadic SOU)
- **[Levy, Schmidt-Schauß, Villaret 2004]** NP-completeness of Monadic SOU

CU and LSOU:

- **[Comon 1993 and Schmidt-Schauss 1995]** introduction of CU
- **[Levy 1996]** Complete LSOU procedure
- **[de Groote 2000]** Decidability of Linear HO Matching
- **[Jez 2014]** Decidability of CU

BSOU:

General Un/Decidability Results

General SOU:

- [Gould 1966] Decidability of SO Matching
- [Lucchesi 1972 and Huet 1973] Third-Order Unification is undecidable
- [Pietrzykowski 1973] Complete SOU procedure
- [Jensen and Pietrzykowski 1976] Complete HOU procedure
- [Goldfarb 1981] SOU is undecidable
- [Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)
- [Levy, Schmidt-Schauß, Villaret 2004] NP-completeness of Monadic SOU

CU and LSOU:

- [Comon 1993 and Schmidt-Schauss 1995] introduction of CU
- [Levy 1996] Complete LSOU procedure
- [de Groote 2000] Decidability of Linear HO Matching
- [Jez 2014] Decidability of CU

BSOU:

General SOU:

- **[Gould 1966]** Decidability of SO Matching
- **[Lucchesi 1972 and Huet 1973]** Third-Order Unification is undecidable
- **[Pietrzykowski 1973]** Complete SOU procedure
- **[Jensen and Pietrzykowski 1976]** Complete HOU procedure
- **[Goldfarb 1981]** SOU is undecidable

CU and LSOU:

- **[Comon 1993 and Schmidt-Schauss 1995]** introduction of CU
- **[Levy 1996]** Complete LSOU procedure
- **[de Groote 2000]** Decidability of Linear HO Matching
- **[Jez 2014]** Decidability of CU

BSOU:

- **[Schmidt-Schauß 2004]** Decidability of BSOU
- **[Levy, Schmidt-Schauß, Villaret 2006]** NP-completeness of BSOU

SO Pre-Unification [Huet 1975]

Simplif.: $\{\mathbf{f}(t_1,\ldots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\ldots,u_n)\} \cup E \Rightarrow$
$\{t_1 \stackrel{?}{=} u_1,\ldots,t_n \stackrel{?}{=} u_n\} \cup E$

Imitation: $\{\mathbf{X}(t_1,\ldots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\ldots,u_m)\} \cup E \Rightarrow$
$\Big(\{X'(t_1,\ldots,t_n) \stackrel{?}{=} u_1,\ldots,X'(t_1,\ldots,t_n) \stackrel{?}{=} u_m\} \cup E\Big)\,\rho$
$\rho = [X \mapsto \lambda y_1,\ldots,y_n\,.\,f(X'_1(y_1,\ldots,y_n),\ldots,X'_m(y_1,\ldots,y_n))]$

Projection: $\{\mathbf{X}(t_1,\ldots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\ldots,u_m)\} \cup E \Rightarrow$
$\Big(\{t_i \stackrel{?}{=} f(u_1,\ldots,u_m)\} \cup E\Big)\,\rho$
$\rho = [X \mapsto \lambda y_1,\ldots,y_n\,.\,y_i]$

Flex-flex equations trivially solvable:
$\{X_1(\ldots) \stackrel{?}{=} Y_1(\ldots),\ldots,X_n(\ldots) \stackrel{?}{=} Y_n(\ldots)\}$
Intantiate $X_i, Y_i \mapsto \lambda x_1,\ldots,x_n.a$

BSO Pre-Unification

Simplif.: $\{\mathbf{f}(t_1,\dots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\dots,u_n)\} \cup E \Rightarrow$
$\{t_1 \stackrel{?}{=} u_1,\dots,t_n \stackrel{?}{=} u_n\} \cup E$

Imitation: $\{\mathbf{X}(t_1,\dots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\dots,u_m)\} \cup E \Rightarrow$
$\left(\{X'(t_{\pi(1)},\dots,t_{\pi(r)}) \stackrel{?}{=} u_1,\dots,X'(t_{\pi(s)},\dots,t_{\pi(n)}) \stackrel{?}{=} u_m\} \cup E\right)\rho$
$\rho = [X \mapsto \lambda y_1,\dots,y_n\,.\,f(X'_1(y_{\pi(1)},\dots,y_{\pi(r)}),\dots,X'_m(y_{\pi(s)},\dots,y_{\pi(n)}))]$
for some permutation π

Projection: $\{\mathbf{X}(t_1,\dots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\dots,u_m)\} \cup E \Rightarrow$
$\left(\{t_i \stackrel{?}{=} f(u_1,\dots,u_m)\} \cup E\right)\rho$
$\rho = [X \mapsto \lambda y_1,\dots,y_n\,.\,y_i]$

Flex-flex equations trivially solvable

LSO Unification [Levy 1996]

Simplif.: $\{\mathbf{f}(t_1,\dots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\dots,u_n)\} \cup E \Rightarrow$
$\{t_1 \stackrel{?}{=} u_1,\dots,t_n \stackrel{?}{=} u_n\} \cup E$

Imitation: $\{\mathbf{X}(t_1,\dots,t_n) \stackrel{?}{=} \mathbf{f}(u_1,\dots,u_m)\} \cup E \Rightarrow$

$$\Big(\{X'(t_{\pi(1)},\dots,t_{\pi(r)}) \stackrel{?}{=} u_1,\dots,X'(t_{\pi(s)},\dots,t_{\pi(n)}) \stackrel{?}{=} u_m\} \cup E\Big)\,\rho$$

$$\rho = [X \mapsto \lambda y_1,\dots,y_n \,.\, f(X'_1(y_{\pi(1)},\dots,y_{\pi(r)}),\dots,X'_m(y_{\pi(s)},\dots,y_{\pi(n)}))]$$

for some permutation π

Projection: $\{\mathbf{X}(t) \stackrel{?}{=} \mathbf{f}(u_1,\dots,u_m)\} \cup E \Rightarrow \Big(\{t \stackrel{?}{=} f(u_1,\dots,u_m)\} \cup E\Big)\,\rho$

$$\rho = [X \mapsto \lambda y \,.\, y]$$

Flex-flex: $\{\mathbf{X}(t_1,\dots,t_n) \stackrel{?}{=} \mathbf{Y}(u_1,\dots,u_m)\} \cup E \Rightarrow$

$$\Big(\{F'_1(t_{\pi(1)},\dots) \stackrel{?}{=} u_{\tau(1)},\dots,t_{\pi(r)} \stackrel{?}{=} G'_1(u_{\tau(s)},\dots) \cup E\Big)\,\rho$$

$$\begin{aligned}\rho = \quad & X \mapsto \lambda y_1,\dots,y_n \,.\, H(F'_1(y_{\pi(1)},\dots) \quad ,\dots, \quad y_{\pi(r)} \quad ,\dots)\\ & Y \mapsto \lambda z_1,\dots,z_m \,.\, H(z_{\tau(1)} \quad ,\dots, \quad G'_1(z_{\tau(s)},\dots) \quad ,\dots)\end{aligned}$$

To Imitate or To Project?

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

Boolean variable $\mathbf{x} \longrightarrow$ SO variable $\mathbf{X}$

$$\begin{array}{lcl} X \mapsto \lambda x \,.\, f(x) & \longrightarrow & x \text{ is true} \\ X \mapsto \lambda x \,.\, x & \longrightarrow & x \text{ is false} \end{array}$$

$$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{a}))) \stackrel{?}{=} \mathbf{f}(\mathbf{a})$$

(If necessary) force variables to use their argument:
$\mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{b}))) \stackrel{?}{=} \mathbf{f}(\mathbf{b})$

This proves NP-hardness of SO matching, Monadic SOU, CU and BSOU

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

1-IN-3SAT:

Clauses with 3 disjunctive variables

Satisfiable if exists an assignment making true **just one** variable in each clause

Boolean variable $\mathbf{x}$ $\longrightarrow$ SO variable $\mathbf{X}$

$$\begin{array}{lcl} X \mapsto \lambda x\,.\,f(x) & \longrightarrow & x \text{ is true} \\ X \mapsto \lambda x\,.\,x & \longrightarrow & x \text{ is false} \end{array}$$

$$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{a}))) \stackrel{?}{=} \mathbf{f}(\mathbf{a})$$

To Imitate or To Project?

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

Boolean variable **x** $\longrightarrow$ SO variable **X**

$$\begin{array}{lcl} X \mapsto \lambda x \,.\, f(x) & \longrightarrow & x \text{ is true} \\ X \mapsto \lambda x \,.\, x & \longrightarrow & x \text{ is false} \end{array}$$

$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{a}))) \stackrel{?}{=} \mathbf{f}(\mathbf{a})$

(If necessary) force variables to use their argument:
$\mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{b}))) \stackrel{?}{=} \mathbf{f}(\mathbf{b})$

This proves NP-hardness of SO matching, Monadic SOU, CU and BSOU

To Imitate or To Project?

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

Boolean variable **x** $\longrightarrow$ SO variable **X**

$$\begin{array}{lcl} X \mapsto \lambda x \,.\, f(x) & \longrightarrow & x \text{ is true} \\ X \mapsto \lambda x \,.\, x & \longrightarrow & x \text{ is false} \end{array}$$

$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X(Y(Z(a)))} \stackrel{?}{=} \mathbf{f(a)}$

(If necessary) force variables to use their argument:
$\mathbf{X(Y(Z(b)))} \stackrel{?}{=} \mathbf{f(b)}$

This proves NP-hardness of SO matching, Monadic SOU, CU and BSOU

To Imitate or To Project?

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

Boolean variable $\mathbf{x} \longrightarrow$ SO variable $\mathbf{X}$

$$\begin{array}{lcl} X \mapsto \lambda x\,.\,f(x) & \longrightarrow & x \text{ is true} \\ X \mapsto \lambda x\,.\,x & \longrightarrow & x \text{ is false} \end{array}$$

$$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X(Y(Z(a)))} \stackrel{?}{=} \mathbf{f(a)}$$

(If necessary) force variables to use their argument:
$\mathbf{X(Y(Z(b)))} \stackrel{?}{=} \mathbf{f(b)}$

This proves NP-hardness of SO matching, Monadic SOU, CU and BSOU

To Imitate or To Project?

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

Boolean variable **x** $\longrightarrow$ SO variable **X**

$$
\begin{array}{lcl}
X \mapsto \lambda x\,.\,f(x) & \longrightarrow & x \text{ is true} \\
X \mapsto \lambda x\,.\,x & \longrightarrow & x \text{ is false}
\end{array}
$$

$$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X(Y(Z(a)))} \stackrel{?}{=} \mathbf{f(a)}$$

(If necessary) force variables to use their argument:
$\mathbf{X(Y(Z(b)))} \stackrel{?}{=} \mathbf{f(b)}$

This proves NP-hardness of SO matching, Monadic SOU, CU and BSOU

To Imitate or To Project?

Source of **NP-hardness**:

Reduce: **1-IN-3SAT** $\longrightarrow$ **SOU**

Boolean variable $\mathbf{x} \longrightarrow$ SO variable $\mathbf{X}$

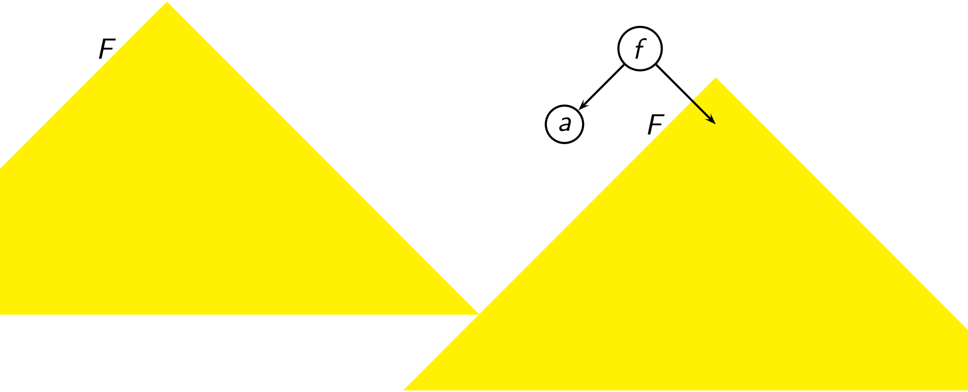
$$\begin{array}{lcl} X \mapsto \lambda x \,.\, f(x) & \longrightarrow & x \text{ is true} \\ X \mapsto \lambda x \,.\, x & \longrightarrow & x \text{ is false} \end{array}$$

$$\mathbf{x} \vee \mathbf{y} \vee \mathbf{z} \quad \longrightarrow \quad \mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{a}))) \stackrel{?}{=} \mathbf{f}(\mathbf{a})$$

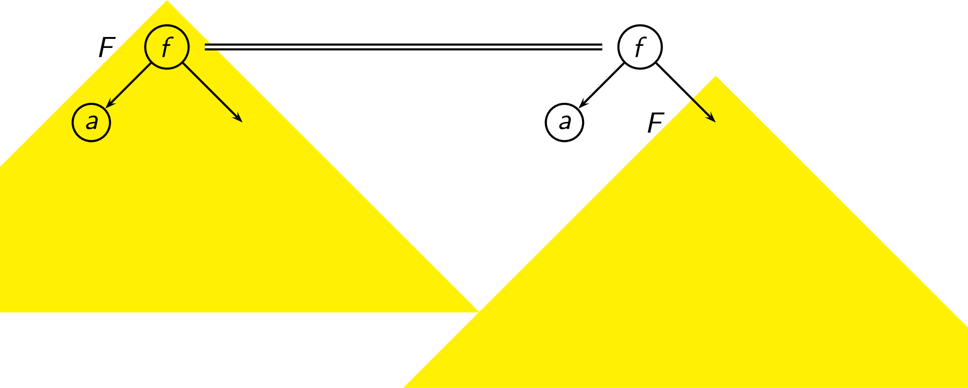
(If necessary) force variables to use their argument:
$\mathbf{X}(\mathbf{Y}(\mathbf{Z}(\mathbf{b}))) \stackrel{?}{=} \mathbf{f}(\mathbf{b})$

This proves NP-hardness of SO matching, Monadic SOU, CU and BSOU

Non-Termination of Imitation

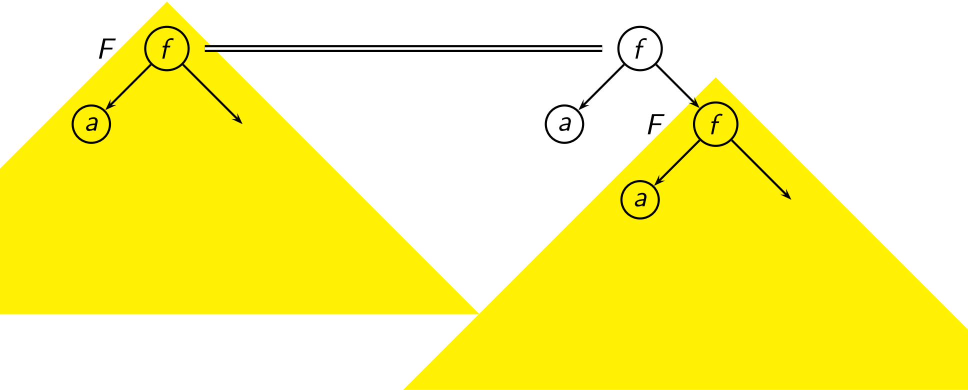


Non-Termination of Imitation



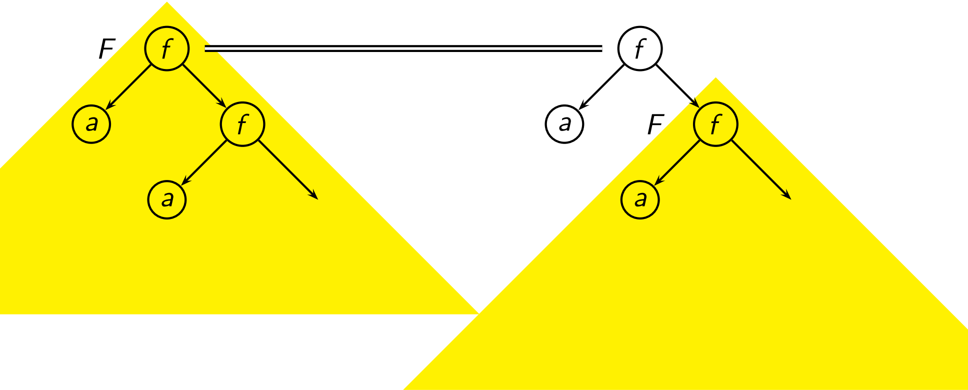
$$F \mapsto \lambda x\,.\,f(a, \ldots)$$

Non-Termination of Imitation



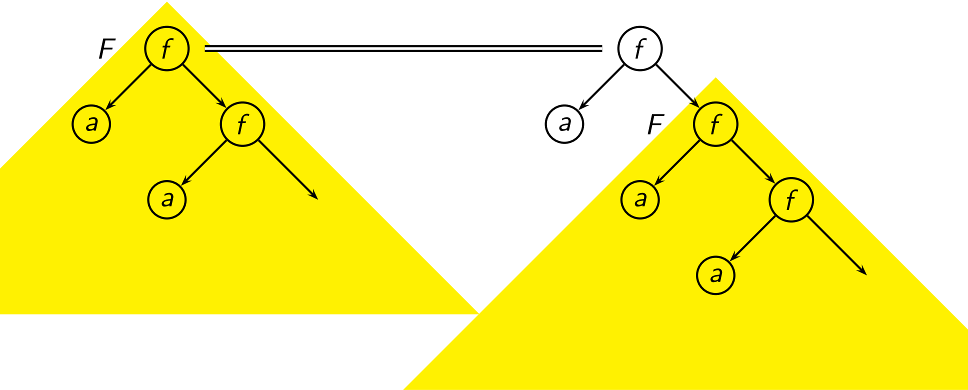
$$F \mapsto \lambda x \,.\, f(a, \dots)$$

Non-Termination of Imitation



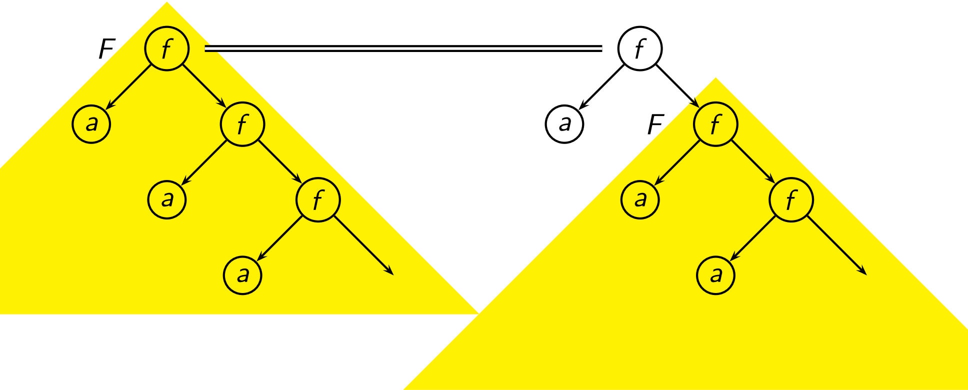
$$F \mapsto \lambda x \,.\, f(a, f(a, \ldots))$$

Non-Termination of Imitation

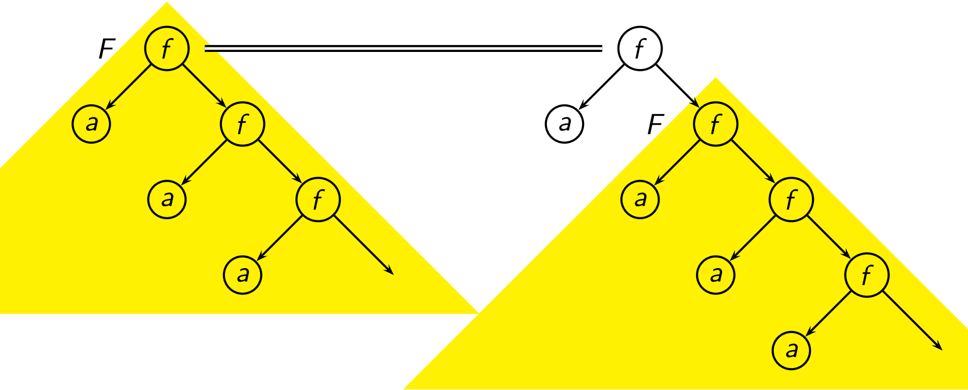


$$F \mapsto \lambda x \,.\, f(a, f(a, \ldots))$$

Non-Termination of Imitation

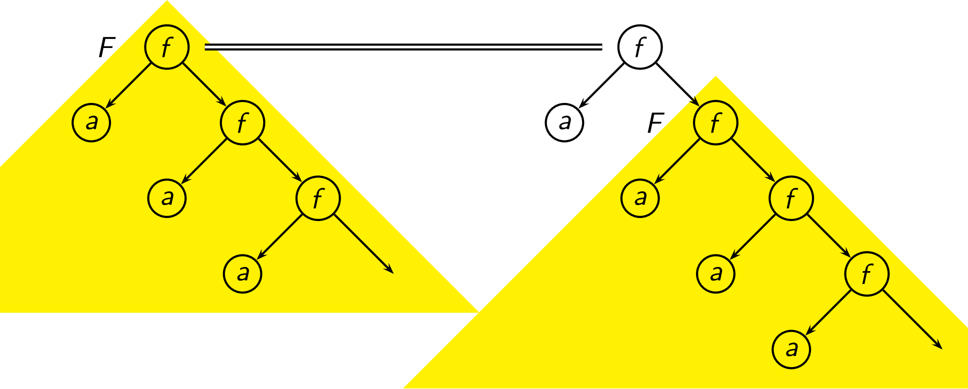


$$F \mapsto \lambda x \,.\, f(a, f(a, f(a, \ldots)))$$



$$F \mapsto \lambda x\,.\, f(a, f(a, f(a, \ldots)))$$

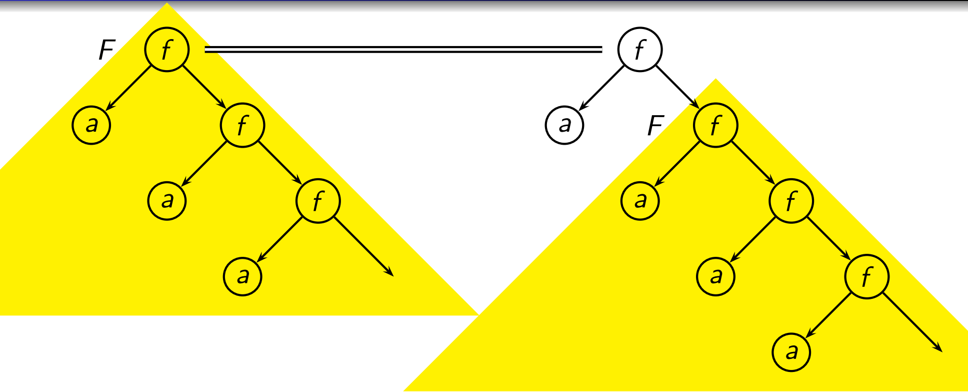
Non-Termination of Imitation



$$F \mapsto \lambda x\,.\,f(a, f(a, f(a, \ldots)))$$

$$F \mapsto \lambda x\,.\,[f(a, \bullet)]^n \ldots$$

Non-Termination of Imitation



$$F \mapsto \lambda x\,.\,[f(a,\bullet)]^n \ldots$$

This already happens in Word Unification

$$X \ldots \stackrel{?}{=} a \cdot X \ldots$$

$$X \mapsto a^n \ldots$$

Exponent of Periodicity

$$F \mapsto \lambda x\,.\,[f(a,\bullet)]^n \ldots$$

This already happens in Word Unification

$$X \ldots \stackrel{?}{=} a \cdot X \ldots$$

$$X \mapsto a^n \ldots$$

We can limit the value of n without affecting solvability

Lemma (SchmidtSchauß 2004)

For every problem E, every size-minimal unifier σ, and every variable X, if C^n is a nonempty subcontext of $\sigma(X)$, then $n \leq \mathcal{O}(2^{|E|^2})$.

$$F \mapsto \lambda x\,.\,[f(a,\bullet)]^n \ldots$$

This already happens in Word Unification

$$X \ldots \stackrel{?}{=} a \cdot X \ldots$$

$$X \mapsto a^n \ldots$$

We can limit the value of n without affecting solvability

Lemma (SchmidtSchauß 2004)

For every problem E, every size-minimal unifier σ, and every variable X, if C^n is a nonempty subcontext of $\sigma(X)$, then $n \leq \mathcal{O}(2^{|E|^2})$.

Monadic SOU

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)

[Farmer 1991] SOU is undecidable even if SO variables are unary

Most-general/size-minimal solutions only use constants occurring in the problem

Even if all variables are unary, we may need n-ary variables

Restrict variables to be unary is not a problem

[Levy Villaret 2002] SOU, BSOU and LSOU can be reduced to their restricted form with **just one binary** symbol

Monadic SOU

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)

[Farmer 1991] SOU is undecidable even if SO variables are unary

Most-general/size-minimal solutions only use constants occurring in the problem

Hint: Replace non-original constants by fresh variables

Even if all variables are unary, we may need n-ary variables

Restrict variables to be unary is not a problem

[Levy Villaret 2002] SOU, BSOU and LSOU can be reduced to their restricted form with **just one binary** symbol

Monadic SOU

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)

[Farmer 1991] SOU is undecidable even if SO variables are unary

Most-general/size-minimal solutions only use constants occurring in the problem

Even if all variables are unary, we may need n-ary variables

Hint: Consider $X(a) \stackrel{?}{=} Y(b)$ and
solution $X \mapsto \lambda x \,.\, Z(x, b)$, $Y \mapsto \lambda y \,.\, Z(a, y)$

This is a problem when representing most general solutions of CU

Restrict variables to be unary is not a problem

[Levy Villaret 2002] SOU, BSOU and LSOU can be reduced to their restricted form with **just one binary** symbol

Monadic SOU

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)

[Farmer 1991] SOU is undecidable even if SO variables are unary

Most-general/size-minimal solutions only use constants occurring in the problem

Even if all variables are unary, we may need n-ary variables

Restrict variables to be unary is not a problem

[Levy Villaret 2002] SOU, BSOU and LSOU can be reduced to their restricted form with **just one binary** symbol

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)
[Farmer 1991] SOU is undecidable even if SO variables are unary
Most-general/size-minimal solutions only use constants occurring in the problem
Even if all variables are unary, we may need n-ary variables
Restrict variables to be unary is not a problem

$$h(X(g(a), Z)) \stackrel{?}{=} Y(b, f(X(c, d)))$$

...consider the most general unifier:

$$h(U(T(g(a), b, f(U(T(c, a, f(U(d))))))))$$
$$h(U(T(g(a), b, f(U(T(c, a, f(U(d))))))))$$

...by instantiating $T \mapsto \lambda x, y, z\,.\,T'(z)$:

$$h(U(T'(f(U(T'(f(U(d)))))))) = h(U(T'(f(U(T'(f(U(d))))))))$$

...we obtain a solution of the problem instantiated by

$$\begin{array}{lcl} X & \mapsto & \lambda x, y\,.\,X'(y) \\ Y & \mapsto & \lambda x, y\,.\,Y'(y) \end{array}$$

Monadic SOU

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)

[Farmer 1991] SOU is undecidable even if SO variables are unary

Most-general/size-minimal solutions only use constants occurring in the problem

Even if all variables are unary, we may need n-ary variables

Restrict variables to be unary is not a problem

[Levy Villaret 2002] SOU, BSOU and LSOU can be reduced to their restricted form with **just one binary** symbol

Hint: Translate
$$\begin{array}{lcl} f(t_1,\dots,t_n) & \longrightarrow & @(\dots @(f,t_1)\dots,t_n) \\ X(t) & \longrightarrow & X(t) \end{array}$$

(a sort of partial curryfication)

This reduction is correct if variables do not "touch" like in $X(Y(t))$

We can guess head symbol in Y and avoid this situation

Monadic SOU

[Farmer 1988] SOU is decidable if all function symbols are unary (Monadic SOU)

[Farmer 1991] SOU is undecidable even if SO variables are unary

Most-general/size-minimal solutions only use constants occurring in the problem

Even if all variables are unary, we may need n-ary variables

Restrict variables to be unary is not a problem

[Levy Villaret 2002] SOU, BSOU and LSOU can be reduced to their restricted form with **just one binary** symbol

[Levy Schmidt-Schauß Villaret 2004] Monadic SOU is NP-complete

Monadic SOU is in NP [Levy Schmidt-Schauß Villaret 2004]

- Represent one of the solutions in polynomial space
- Prove that we can check if a substitution is a solution in polynomial time on this representation

Monadic SOU is in NP [Levy Schmidt-Schauß Villaret 2004]

- Represent one of the solutions in polynomial space
- Prove that we can check if a substitution is a solution in polynomial time on this representation
- Use the exponent of periodicity of size-minimal solutions (we can represent exponents in linear space) **[Makanin, Kościelski and Pacholski]**
- Use (singleton) context free grammars to represent solutions
- Given two singleton CFG we can check if they define the same word in polynomial time **[Plandowski]**

Monadic SOU is in NP (Some Details)

Lemma

If G defines $w_1, \ldots, w_n$, exists $G' \supseteq G$ defining $w = w_1 \ldots w_n$ s.t.

$$\begin{aligned} |G'| &\leq |G| + n - 1 \\ \mathtt{depth}(G') &\leq \mathtt{depth}(G) + \lceil \log n \rceil \end{aligned}$$

Lemma

If G defines w, for any n, exists $G' \supseteq G$ defining w^n s.t.

$$\begin{aligned} |G'| &\leq |G| + 2\lfloor \log n \rfloor \\ \mathtt{depth}(G') &\leq \mathtt{depth}(G) + \lceil \log n \rceil \end{aligned}$$

Lemma

If G defines w, for any $w' \prec w$, exists $G' \supseteq G$ defining w' s.t.

$$\begin{aligned} |G'| &\leq |G| + \mathtt{depth}(G) \\ \mathtt{depth}(G') &= \mathtt{depth}(G) \end{aligned}$$

Monadic SOU is in NP (Some Details)

Define if $X\, w_1 \stackrel{?}{=} Y\, w_2$ then $X \approx Y$

- One node for every $\approx$-equivalence class of variables.
- For every $X\, w_1 \stackrel{?}{=} a_1 \cdots a_n\, Y\, w_2$

$$L_1 \xrightarrow{a_1} \emptyset \xrightarrow{a_2} \emptyset \quad \cdots \quad \emptyset \xrightarrow{a_n} L_2$$

  where $X \in L_1$ and $Y \in L_2$

- For every $X\, w_1 \stackrel{?}{=} a_1 \cdots a_n\, b$

$$L \xrightarrow{a_1} \emptyset \xrightarrow{a_2} \emptyset \quad \cdots \quad \emptyset \xrightarrow{a_n} b$$

  where $X \in L$

Monadic SOU is in NP (Some Details)

Theorem

Let σ be a size-minimal lazy unifier of $\langle E, G \rangle$ with exponent not exceeding k. Then exist X and G', deriving $\sigma(X)$ and s.t.

$$|G'| \leq |G| + \mathcal{O}(|E|^2\, \mathtt{depth}(G) + \log k)$$
$$\mathtt{depth}(G') \leq \mathtt{depth}(G) + \mathcal{O}(\log k + \log |E|)$$

Theorem

For any solvable equations $\langle E, \emptyset \rangle$, exists a lazy unifier $\langle \sigma, G \rangle$ s.t.

$$|\sigma| = \mathcal{O}(|E|)$$
$$|G| = \mathcal{O}(|E|^5)$$
$$\mathtt{depth}(G) = \mathcal{O}(|E|^2)$$

Two Occurrences per Variable [Levy 1998]

Word Unification is trivially decidable when variables do not occur more than twice:

Imitation: $\{X \cdot w_1 \stackrel{?}{=} a \cdot w_2\} \cup E\} \Rightarrow \Big(\{X' \cdot w_1 \stackrel{?}{=} w_2\} \cup E\}\Big)\,\rho$
where $\rho = [X \mapsto a \cdot X']$

Flex-flex: $\{X \cdot w_1 \stackrel{?}{=} Y \cdot w_2\} \cup E\} \Rightarrow \Big(\{X' \cdot w_1 \stackrel{?}{=} w_2\} \cup E\}\Big)\,\rho$
where $\rho = [X \mapsto Y \cdot X']$

[Levy 1996] LSOU is decidable when variables do not occur more than twice

[Levy 1998] SOU is undecidable even when variables do not occur more than twice

$$t_1 \to u_1, \ldots, t_m \to u_m \vdash v \to w$$
$$\Updownarrow$$
$$X(f(a,v),u_1,\ldots,u_m) \stackrel{?}{=} f(X(a,t_1,\ldots,t_m),w) \text{ solvable}$$

Two Occurrences per Variable [Levy 1998]

Word Unification is trivially decidable when variables do not occur more than twice:

[Levy 1996] LSOU is decidable when variables do not occur more than twice

[Levy 1998] SOU is undecidable even when variables do not occur more than twice

$$t_1 \to u_1, \dots, t_m \to u_m \vdash v \to w$$

$$\Updownarrow$$

$$X(f(a,v), u_1, \dots, u_m) \stackrel{?}{=} f(X(a, t_1, \dots, t_m), w) \text{ solvable}$$

f and a are symbols not used in the rewriting system

Two Occurrences per Variable [Levy 1998]

Word Unification is trivially decidable when variables do not occur more than twice:
[Levy 1996] LSOU is decidable when variables do not occur more than twice
[Levy 1998] SOU is undecidable even when variables do not occur more than twice

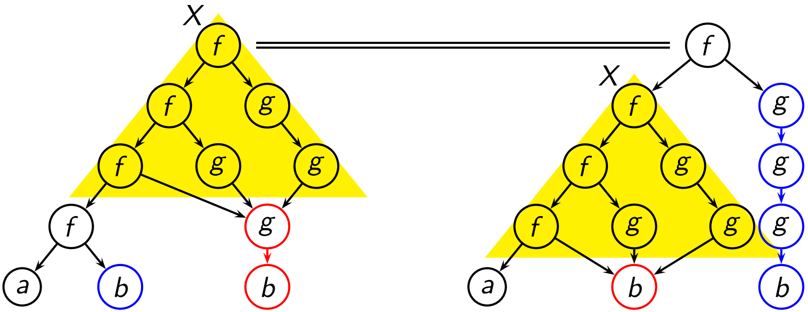
$$t_1 \to u_1, \dots, t_m \to u_m \vdash v \to w$$
$$\Updownarrow$$
$$X(f(a,v), u_1, \dots, u_m) \stackrel{?}{=} f(X(a, t_1, \dots, t_m), w) \text{ solvable}$$

f and a are symbols not used in the rewriting system

$$t_1 \to u_1, \dots, t_m \to u_m \vdash v \to w$$
$$\Updownarrow$$
$$X(f(a,v), u_1, \dots, u_m) \stackrel{?}{=} f(X(a, t_1, \dots, t_m), w) \text{ solvable}$$

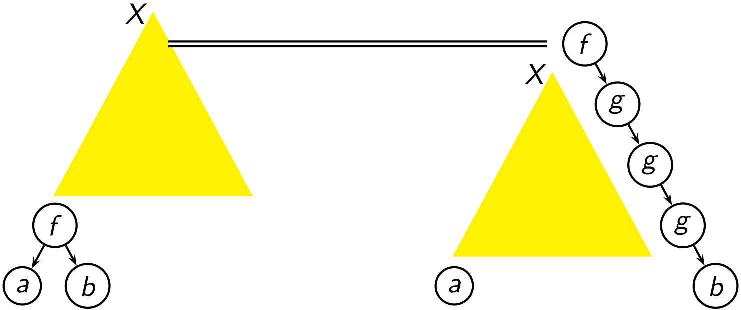
Example:

$$b \to g(b) \vdash b \to g(g(g(b)))$$



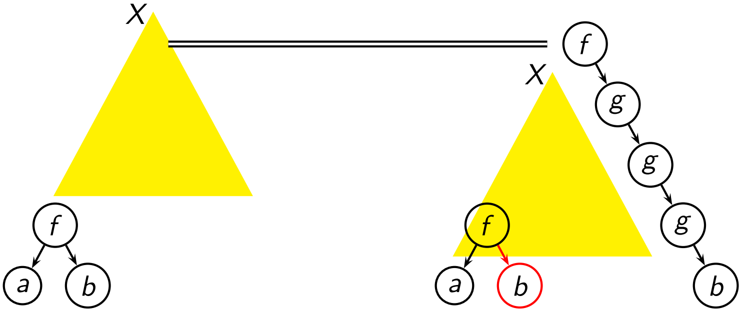
Encoding Sequences

$$b \to g(b) \vdash$$



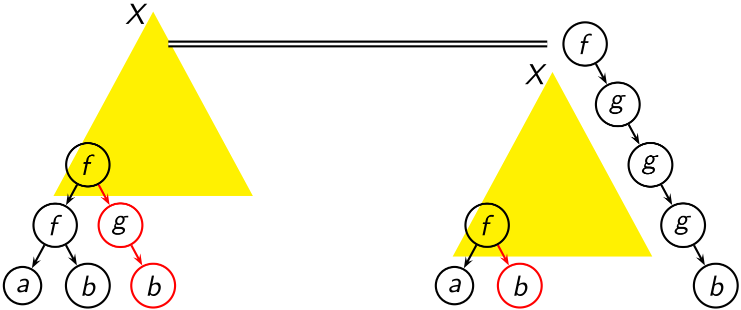
Encoding Sequences

$$b \to g(b) \vdash b$$

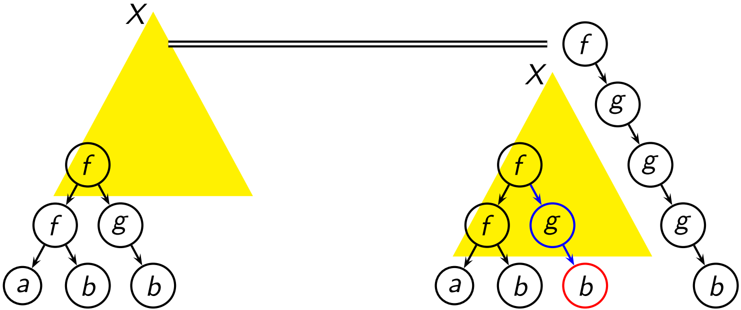


Encoding Sequences

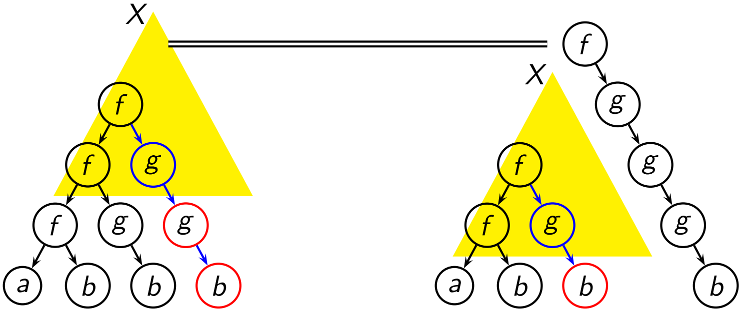
$$b \to g(b) \vdash b \to g(b)$$



$$b \to g(b) \vdash b \to g(b)$$

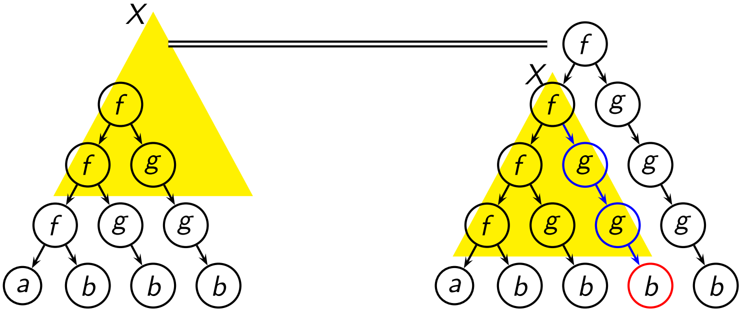


$$b \rightarrow g(b) \vdash b \rightarrow g(b) \rightarrow g(g(b))$$

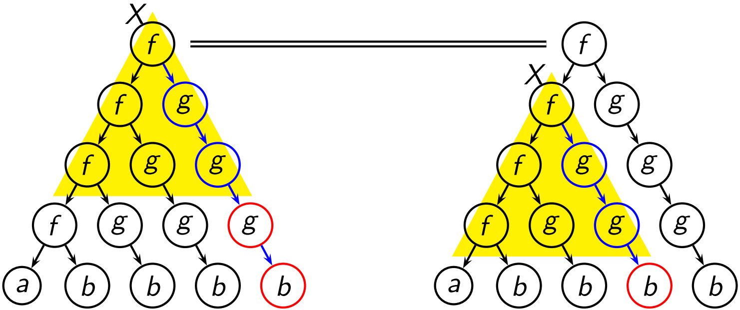


Encoding Sequences

$$b \to g(b) \vdash b \to g(b) \to g(g(b))$$



$$b \to g(b) \vdash b \to g(b) \to g(g(b)) \to g(g(g(b)))$$



Just One Variable

[Levy Veanes 2000] SOU is reducible to SOU with just one variable and the same (plus one) number of occurrences

$$\bigcup_{1\le i\le m}\ \bigcup_{1\le j\le k_i} F_i(t_{ij}^1,\dots,t_{ij}^n) \stackrel{?}{=} u_{ij},$$

$$\Updownarrow$$

$$\bigcup_{1\le i\le m}\ \bigcup_{1\le j\le k_i} G(t_{ij}^1,\dots,t_{ij}^n) \stackrel{?}{=} g(\underbrace{\_,\dots,\_}_{i-1},u_{ij},\underbrace{\_,\dots,\_}_{m-i}),$$

where G is a fresh variable,
g an appropriate constant, and
"$\_$" denotes fresh and distinct first-order variables

Plus $G(c,\dots,c) \stackrel{?}{=} g(\_,\dots,\_)$, if some t_{ij}^k is a variable

Just One Variable

$$\begin{aligned} F_1(t_1, u_1) &\stackrel{?}{=} u_1 \\ F_1(t_2, u_2) &\stackrel{?}{=} u_2 \\ F_2(t_3, u_3) &\stackrel{?}{=} u_3 \\ F_2(t_4, u_4) &\stackrel{?}{=} u_4 \end{aligned} \tag{1}$$

$$\begin{aligned} G(t_1, u_1) &\stackrel{?}{=} g(u_1, X_1) \\ G(t_2, u_2) &\stackrel{?}{=} g(u_2, X_2) \\ G(t_3, u_3) &\stackrel{?}{=} g(X_3, u_3) \\ G(t_4, u_4) &\stackrel{?}{=} g(X_4, u_4) \end{aligned} \tag{2}$$

$\Rightarrow$ If σ solves (1), then

$$\begin{aligned} G &\mapsto x, y\,.\,g(F_1(x, y), F_2(x, y)) \\ X_1 &\mapsto F_2(t_1, u_1) \\ X_2 &\mapsto F_2(t_2, u_2) \\ X_3 &\mapsto F_1(t_3, u_3) \\ X_4 &\mapsto F_1(t_4, u_4) \end{aligned}$$

Just One Variable

$$
\begin{aligned}
F_1(t_1, u_1) &\stackrel{?}{=} u_1 \\
F_1(t_2, u_2) &\stackrel{?}{=} u_2 \\
F_2(t_3, u_3) &\stackrel{?}{=} u_3 \\
F_2(t_4, u_4) &\stackrel{?}{=} u_4
\end{aligned}
\tag{1}
$$

$$
\begin{aligned}
G(t_1, u_1) &\stackrel{?}{=} g(u_1, X_1) \\
G(t_2, u_2) &\stackrel{?}{=} g(u_2, X_2) \\
G(t_3, u_3) &\stackrel{?}{=} g(X_3, u_3) \\
G(t_4, u_4) &\stackrel{?}{=} g(X_4, u_4)
\end{aligned}
\tag{2}
$$

$\Leftarrow$ If σ solves (2), and g is "inside" $\sigma(G)$, then the imitation step $G \mapsto \lambda x, y \,.\, g(G_1(x, y), G_2(x, y))$ transforms (2) in (1)

Just One Variable

The equation $G(c,\dots,c) \stackrel{?}{=} g(\_,\dots,\_)$ may be necessary

$$\begin{aligned} F_1(X) &\stackrel{?}{=} a \\ F_1(Y) &\stackrel{?}{=} b \\ F_2(X) &\stackrel{?}{=} b \\ F_2(Y) &\stackrel{?}{=} a \end{aligned} \tag{1}$$

$$\begin{aligned} G(X) &\stackrel{?}{=} g(a, X_1) \\ G(Y) &\stackrel{?}{=} g(b, X_2) \\ G(X) &\stackrel{?}{=} g(X_3, b) \\ G(Y) &\stackrel{?}{=} g(X_4, a) \end{aligned} \tag{2}$$

(1) is unsolvable
(2) is solvable

$$\begin{aligned} G &\mapsto \lambda x\,.\,x \\ X &\mapsto g(a, b) \\ Y &\mapsto g(b, a) \end{aligned}$$

Ground Arguments

Encode execution of a (Universal) Turing Machine as a sequence of pairs of states

$$((v_1, v_1^+), (v_2, v_2^+), \dots, (v_k, v_k^+)$$

Encode execution of a (Universal) Turing Machine as a sequence of pairs of states

$$((v_1, v_1^+), (v_2, v_2^+), \dots, (v_k, v_k^+))$$

Use 2 equations:

$$F(\bar{t}, f(b, a)) \stackrel{?}{=} f(X, F(\bar{u}, a))$$

ensures $v_{i+i} = v_i^+$
$F(\bar{t}, f(t', a))$ encodes $(v_1, \dots, v_k, b)$
$f(X, F(\bar{u}, a))$ encodes $(X, v_1^+, \dots, v_k^+)$
X encodes the initial state

Ground Arguments

Encode execution of a (Universal) Turing Machine as a sequence of pairs of states

$$((v_1, v_1^+), (v_2, v_2^+), \ldots, (v_k, v_k^+))$$

Use 2 equations:

$$F(\bar{t}, f(b, a)) \stackrel{?}{=} f(X, F(\bar{u}, a))$$

ensures $v_{i+i} = v_i^+$

$$G(\bar{l}, f'(a, a')) \stackrel{?}{=} f'(F(\bar{v}, a), G(\bar{r}, a'))$$

ensures v_i^+ is successor of v_i

$\bar{l}$ and $\bar{r}$ encode the transitions of the Turing Machine

$\bar{t}$, $\bar{u}$ and $\bar{v}$ only depends on the alphabet

Ground Arguments

Encode execution of a (Universal) Turing Machine as a sequence of pairs of states

$$((v_1, v_1^+), (v_2, v_2^+), \ldots, (v_k, v_k^+))$$

Use 2 equations:

$$F(\bar{t}, f(b, a)) \stackrel{?}{=} f(X, F(\bar{u}, a))$$

$$G(\bar{l}, f'(a, a')) \stackrel{?}{=} f'(F(\bar{v}, a), G(\bar{r}, a'))$$

Encodes Halting Problem of Universal TM on input X
Undecidability of SOU for **5** occurrences of **one** second-order variable, even when the variable is only applied to **ground** terms

Thanks for your attention!!