evropský
sociální
fond v ČR

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání
pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vybrané partie z matematických základů informatiky

Petr Jančar

katedra informatiky FEI VŠB-TU Ostrava

přednáška na semináři o výuce matematiky v rámci projektu
MI21 – Matematika pro inženýry 21. století
17. května 2012

Plán

- poznámky k předmětům (mgr. studium)
  - 460-4005/01 - Teoretická informatika (TI)
  - 460-4043/01 - Vybrané partie teoretické informatiky (VPTI)
  - 460-4016/01 - Modelování a verifikace (MaV)
  - 460-4006 - Petriho sítě I (PES I); 460-4019 - Petriho sítě II (PES II)
  - 460-4037/01 - Teorie her (TEH)
- výzkumná práce v oblasti (mezí automatizované) verifikace
  - aktuální výsledek:
    P. Jančar:
    Decidability of DPDA Language Equivalence via First-Order Grammars
    (nový důkaz ekvivalence deterministických zásobníkových automatů)
    (přijato na 27th Annual ACM/IEEE Symposium on
    Logic in Computer Science (LICS 2012), June 2012)

Předmět Teoretická informatika

- Konečné automaty a regulární jazyky.
- Bezkontextové gramatiky (a jazyky), zásobníkové automaty.
- Turingovy stroje, stroje RAM. (Algoritmická) rozhodnutelnost, částečná rozhodnutelnost, nerozhodnutelnost.
- Složitost algoritmů. Třídy složitosti problémů.
- (LOGSPACE, NLOGSPACE, NC,) PTIME, NPTIME, PSPACE, (....).
- Nástin problematiky aproximačních, pravděpodobnostních, paralelních a distribuovaných algoritmů.

Vybrané partie z teoretické informatiky

- Pravděpodobnostní algoritmy. (RSA šifrovací systém. Randomizovaný protokol.)
- Interaktivní protokoly. Třída IP.
  „Zero-knowledge proofs“. „Probabilistically checkable proofs“.
- Aproximační algoritmy. Výsledky o neaproximovatelnosti.
- Automaty a logiky na nekonečných slovech.

Modelování a verifikace

Verifikace ... lze chápat jako potvrzení pravdivosti nějakého tvrzení (např. „systém se chová v souladu se zadanou specifikací“)

Gottfried Wilhelm Leibniz (1646 - 1716):

> *Kdykoliv existují různé názory na určitá fakta, neměli bychom o nich diskutovat tak, jak to obvykle dělají filozofové; místo toho bychom měli pravdu ‘vypočítat’.*

lingua characteristica (k vyjádření všemožných vlastností)
calculus ratiocinator (pravidla poskytující ‘rozhodovací proceduru’)
(za použití univerzální encyklopedie)

(Leibniz prý plánoval svůj projekt na další tři století)

Verifikace počítačového programu (dělení čísel)

vstupní podmínka $\{\ C_1\colon x_1 \geq 0, x_2 > 0\ \}$ (např. $x_1 = 17$, $x_2 = 5$)

Program P

```
$z_1 := 0; z_2 := x_1;$
**while** $[z_2 \not< x_2]$ **do** $(z_1 := z_1 + 1; z_2 := z_2 - x_2);$
```

$\{\ C_2\colon (x_1 = z_1 \cdot x_2 + z_2) \wedge (0 \leq z_2 < x_2)\ \}$ $(17 = 3 \cdot 5 + 2)$

	x_1, z_1, x_2, z_2	$[\ x_1 = z_1 \cdot x_2 + z_2,$	$0 \leq z_2\ ] \ldots$ INV
s_1:	$17, 0, 5, 17$	$17 = 0 \cdot 5 + 17,$	$0 \leq 17\ (\not< 5)$
s_2:	$17, 1, 5, 12$	$17 = 1 \cdot 5 + 12,$	$0 \leq 12\ (\not< 5)$
s_3:	$17, 2, 5, 7$	$17 = 2 \cdot 5 + 7,$	$0 \leq 7\ (\not< 5)$
s_4:	$17, 3, 5, 2$	$17 = 3 \cdot 5 + 2,$	$0 \leq 2\ (< 5)$

Vygenerované verifikační podmínky:

$\{C_1\}\ z_1 := 0; z_2 := x_1\ \{\mathrm{INV}\}$

$\{\mathrm{INV} \wedge z_2 \not< x_2\}\ z_1 := z_1 + 1; z_2 := z_2 - x_2\ \{\mathrm{INV}\}$

$\{\mathrm{INV} \wedge (y_2 < x_2)\} \Longrightarrow \{C_2\}$

Vývoj programu zároveň s důkazem

David Gries (The science of programming, 1981):

> "the study of program correctness proofs has led to the discovery and elucidation of methods for developing programs. Basically, one attempts to
> develop a program and its proof hand-in-hand,
> with the proof ideas leading the way !"

Donald Knuth . . . literate programming

Některé verifikační nástroje

PVS Specification and Verification System
Stanford research institute, USA http://pvs.csl.sri.com/

The HOL System Cambridge university, UK
http://www.cl.cam.ac.uk/Research/HVG/HOL/

Coq Inria, France http://pauillac.inria.fr/coq/

TLV - Temporal Logic Verifier Weizmann Institute of Science, Israel
http://www.wisdom.weizmann.ac.il/ verify/tlv/

Obsáhlý přehled dostupných verifikačních nástrojů je udržován na Fakultě informatiky MU v Brně, http://anna.fi.muni.cz/yahoda/.

Verifikace ‘jednoduchých’ vlastností

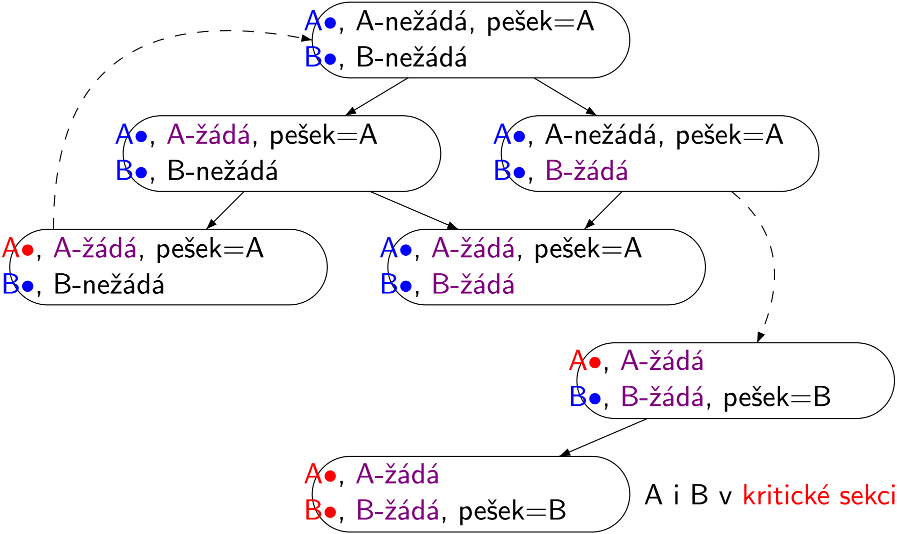
Souběžné, paralelní, distribuované, interaktivní, … systémy
(s ‘nekonečným’ chováním)

(‘Špatný’) příklad: Hymanův algoritmus (CACM, 1966)

```
Process $A$:
Hlavní-Cyklus:
** nekritická sekce **
A-žádá
while pešek $\neq A$ do
        waitfor B-nežádá
        pešek $:= A$
** kritická sekce **
A-nežádá
goto Hlavní-Cyklus
```

```
Process $B$:
Hlavní-Cyklus:
** nekritická sekce **
B-žádá;
while pešek $\neq B$ do
        waitfor A-nežádá
        pešek $:= B$
** kritická sekce **
B-nežádá
goto Hlavní-Cyklus
```

Stavový prostor (Kripkeho struktura)



A i B v kritické sekci !!

Jednoduché (temporální) vlastnosti

bezpečnost (safety properties):

$\forall t : (\neg crit_A(t) \vee \neg crit_B(t))$ $\qquad$ $G(\neg crit_A \vee \neg crit_B)$

("something bad never happens")

živost (liveness properties):

$\forall t : zada_A(t) \Rightarrow (\exists t' : t \leq t' \wedge crit_A(t'))$ $\qquad$ $G(zada_A \Rightarrow F(crit_A))$

"something good eventually happens"

Pro vyjádření (a verifikaci) takových vlastností se osvědčila logika LTL (Linear(-time)Temporal Logic)

Prior 1957,
Amir Pnueli kolem 1977: užití pro verifikaci

Model checking LTL. Logika CTL.

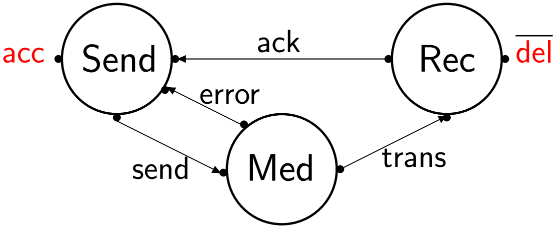
Rozhodni, zda $\boxed{\mathcal{K}, s \models \Phi}$, t.j.,
zda všechny možné (prů)běhy ze stavu s splňují formuli Φ:

- zkonstruuj automat $\mathcal{A}_{\neg\Phi}$, reprezentující všechny běhy nesplňující Φ
- zkonstruuj (produktový) automat $\mathcal{A} = (\mathcal{K}, s) \times \mathcal{A}_{\neg\Phi}$
- otestuj, zda $L(\mathcal{A}) = \emptyset$ (když NE, poskytni protipříklad)

Jinou logikou je CTL (Computation Tree Logic)
(větvící se čas [branching time])

SPIN (LTL model checker)
http://spinroot.com/
Bell Labs
SMV (Symbolic model checking)
http://www-cad.eecs.berkeley.edu/ kenmcmil/smv/
Cadence Berkeley Laboratories

Malý komunikační protokol



rozhraní: acc $\boxed{\text{P}}$ $\overline{\text{del}}$ ..... chování: $\text{Spec} \overset{\text{def}}{=} \text{acc}.\overline{\text{del}}.\text{Spec}$

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\text{Send} & \overset{\text{def}}{=} & \text{acc}.\text{Sending} & \text{Rec} & \overset{\text{def}}{=} & \text{trans}.\text{Del} \\
\text{Sending} & \overset{\text{def}}{=} & \overline{\text{send}}.\text{Wait} & \text{Del} & \overset{\text{def}}{=} & \overline{\text{del}}.\text{Ack} \\
\text{Wait} & \overset{\text{def}}{=} & \text{ack}.\text{Send} + \text{error}.\text{Sending} & \text{Ack} & \overset{\text{def}}{=} & \overline{\text{ack}}.\text{Rec}
\end{array}$$

$$\begin{array}{rcl}
\text{Med} & \overset{\text{def}}{=} & \text{send}.\text{Med}' \\
\text{Med}' & \overset{\text{def}}{=} & \tau.\text{Err} + \overline{\text{trans}}.\text{Med} \\
\text{Err} & \overset{\text{def}}{=} & \overline{\text{error}}.\text{Med}
\end{array}$$

Testování ekvivalence (equivalence checking)

Zde popis specifikace i implementace v CCS
(Calculus of Communicating Systems, Milner)

$$\mathsf{Spec} \overset{\mathrm{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

$$\mathsf{Impl} \overset{\mathrm{def}}{=} (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \smallsetminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

Verifikační otázka: $\boxed{\mathsf{Impl} \overset{?}{\approx} \mathsf{Spec}}$

Vhodnou behaviorální ekvivalencí $\approx$ je zde
(slabá) bisimulační ekvivalence.

Aktuální výzkum

- Böhm, Göller, Jančar: Equivalences on one-counter automata (Concur'10, MFCS'11, příprava časop. verze)
- Czerwinski, Jančar, Kot, Sawa: Bisimilarity between BPA and BPP (třídy procesů generovaných bezkontextovými gramatikami)
- Jančar, Karandikar, Schnoebelen: Unidirectional channel systems can be tested ....
- Forejt, Jančar, Kiefer, Worrell: Bisimilarity and language equivalence of probabilistic pushdown automata ...
- Brázdil, Jančar, Kučera: Reachability Games on Extended Vector Addition Systems with States (ICALP'10, příprava časop. verze)

Language equivalence of deterministic pushdown automata

Example of a (formal) language L over a finite alphabet Σ, so $L \subseteq \Sigma^*$:

$\Sigma = \{\, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, (,)\, \} \cup \{\, \dashv \}$

L … language of arithmetic expressions

e.g., the word (sequence)

$u = \boxed{5 + 28 * (318 - 5 * 24) + 562 \dashv}$ is in L,

$v = \boxed{5 + 28 * (318 - (5 * 24) + 562 \dashv}$ is not in L

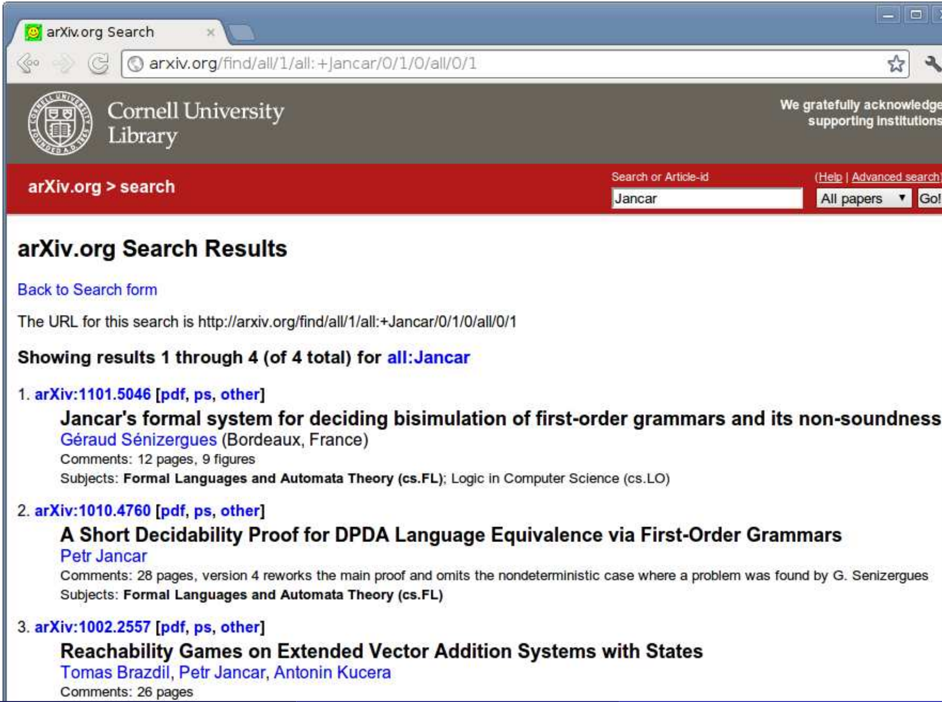
We can view a deterministic pushdown automaton M as a program

- with fixed finite memory; program+memory…finite control unit,
- with a potentially unbounded stack (LIFO, access to the top),
- reading the input word from left-to-right,
- accepting when reading the endmarker $\dashv$ and having the stack empty.

Decidability of $L(M_1) \stackrel{?}{=} L(M_2)$ was open since 1960s (stated in a paper by Ginsburg, Greibach). Another formulation: $L(p\alpha) \stackrel{?}{=} L(q\beta)$ for configurations of the same M (p … control state, α … stack content).

Solution

- Sénizergues G.:
  L(A)=L(B)? Decidability results from complete formal systems.
  Theoretical Computer Science 251(1-2): 1-166 (2001)
  (a preliminary version appeared at ICALP'97; Gödel prize 2002)
- Stirling C.: Decidability of DPDA equivalence.
  Theoretical Computer Science 255, 1-31, 2001
- Sénizergues G.: L(A)=L(B)? A simplified decidability proof.
  Theoretical Computer Science 281(1-2): 555-608 (2002)
- Stirling C.: Deciding DPDA equivalence is primitive recursive.
  ICALP 2002, Lecture Notes in Computer Science 2380, 821-832, Springer 2002 (longer draft paper on the author's web page)
- Sénizergues G.: The Bisimulation Problem for Equational Graphs of Finite Out-Degree.
  SIAM J.Comput., 34(5), 1025–1106 (2005)
  (a preliminary version appeared at FOCS'98)


arXiv.org Search
arxiv.org/find/all/1/all:+Jancar/0/1/0/all/0/1
Cornell University Library
We gratefully acknowledge supporting institutions
arXiv.org > search
Search or Article-id
Jancar
(Help | Advanced search)
All papers
Go!

arXiv.org Search Results

Back to Search form

The URL for this search is http://arxiv.org/find/all/1/all:+Jancar/0/1/0/all/0/1

Showing results 1 through 4 (of 4 total) for all:Jancar

1. **arXiv:1101.5046 [pdf, ps, other]**

   Jancar's formal system for deciding bisimulation of first-order grammars and its non-soundness

   Géraud Sénizergues (Bordeaux, France)

   Comments: 12 pages, 9 figures

   Subjects: **Formal Languages and Automata Theory (cs.FL)**; Logic in Computer Science (cs.LO)

2. **arXiv:1010.4760 [pdf, ps, other]**

   A Short Decidability Proof for DPDA Language Equivalence via First-Order Grammars

   Petr Jancar

   Comments: 28 pages, version 4 reworks the main proof and omits the nondeterministic case where a problem was found by G. Senizergues

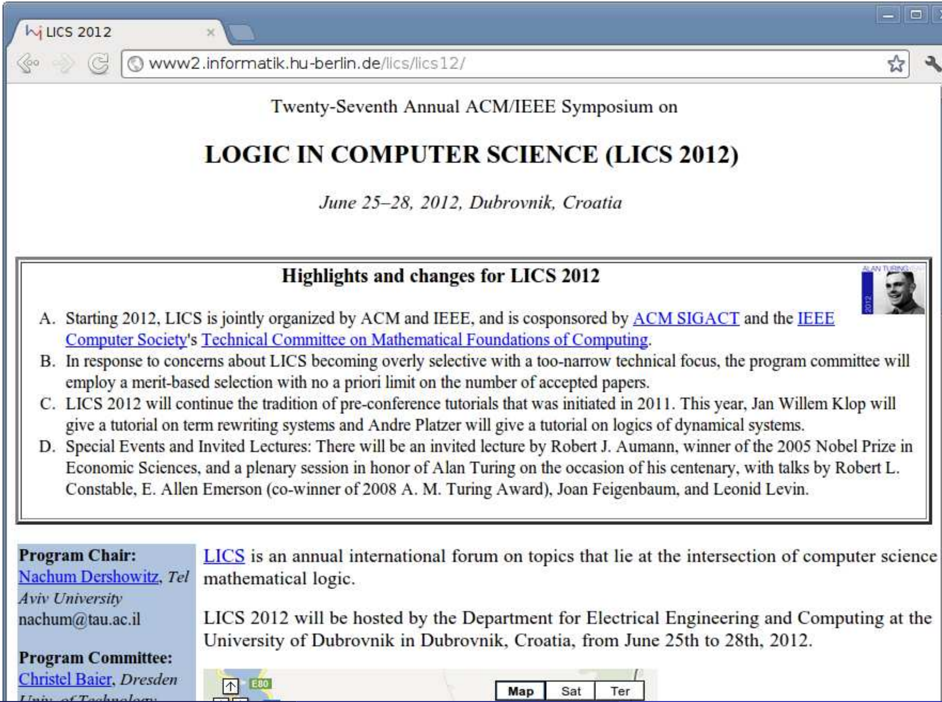
   Subjects: **Formal Languages and Automata Theory (cs.FL)**

3. **arXiv:1002.2557 [pdf, ps, other]**

   Reachability Games on Extended Vector Addition Systems with States

   Tomas Brazdil, Petr Jancar, Antonin Kucera

   Comments: 26 pages



Twenty-Seventh Annual ACM/IEEE Symposium on

LOGIC IN COMPUTER SCIENCE (LICS 2012)

June 25–28, 2012, Dubrovnik, Croatia

Highlights and changes for LICS 2012

A. Starting 2012, LICS is jointly organized by ACM and IEEE, and is cosponsored by ACM SIGACT and the IEEE Computer Society's Technical Committee on Mathematical Foundations of Computing.

B. In response to concerns about LICS becoming overly selective with a too-narrow technical focus, the program committee will employ a merit-based selection with no a priori limit on the number of accepted papers.

C. LICS 2012 will continue the tradition of pre-conference tutorials that was initiated in 2011. This year, Jan Willem Klop will give a tutorial on term rewriting systems and Andre Platzer will give a tutorial on logics of dynamical systems.

D. Special Events and Invited Lectures: There will be an invited lecture by Robert J. Aumann, winner of the 2005 Nobel Prize in Economic Sciences, and a plenary session in honor of Alan Turing on the occasion of his centenary, with talks by Robert L. Constable, E. Allen Emerson (co-winner of 2008 A. M. Turing Award), Joan Feigenbaum, and Leonid Levin.

Program Chair:
Nachum Dershowitz, *Tel Aviv University*
nachum@tau.ac.il

Program Committee:
Christel Baier, *Dresden Univ. of Technology*

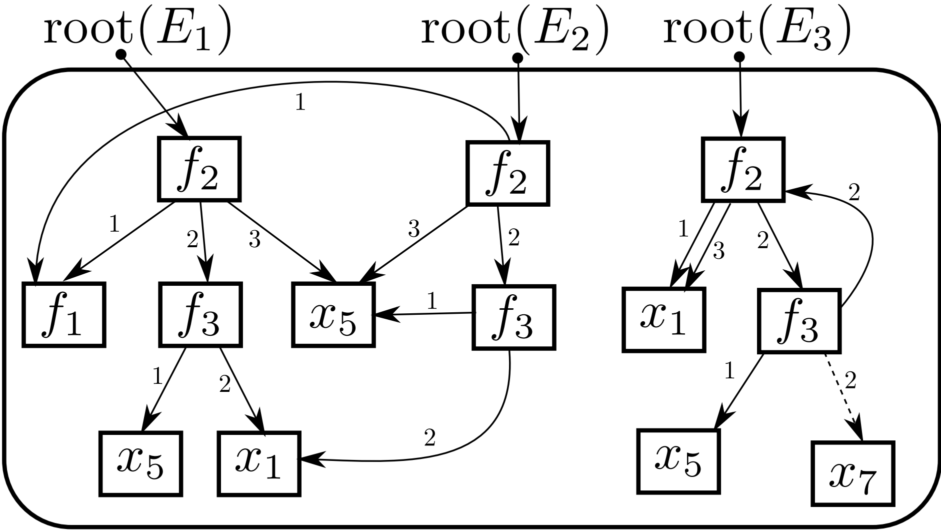
LICS is an annual international forum on topics that lie at the intersection of computer science mathematical logic.

LICS 2012 will be hosted by the Department for Electrical Engineering and Computing at the University of Dubrovnik in Dubrovnik, Croatia, from June 25th to 28th, 2012.

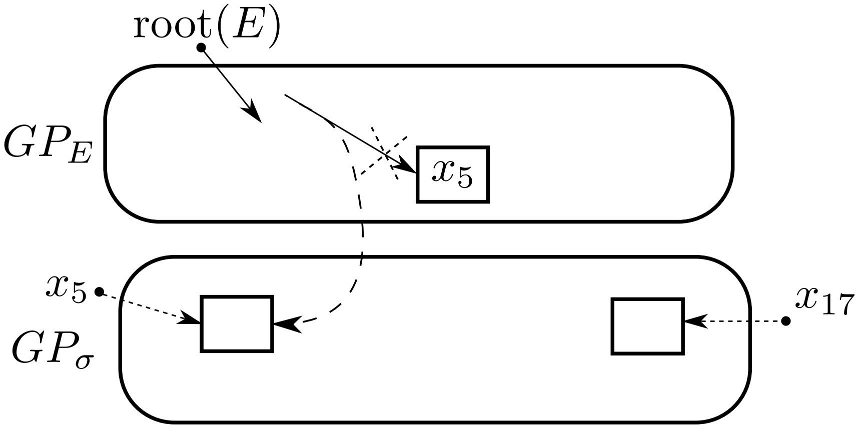
Outline (a short self-contained proof via first-order terms)

- First-order terms, substitutions; <u>regular</u> terms
- (Deterministic) labelled transition systems (LTSs); trace equivalence
- (D)pda configurations as terms; rules as root-rewriting
- LTSs generated by (det-)first-order grammars; semidecidability of nonequivalence
- Simple properties of $\sim$ and its "strata" $\sim_0, \sim_1, \sim_2, \ldots$.
- Algorithm for the positive case based on a Prover-Refuter game
- Soundness of P-R game (obvious)
- Two steps for completeness
  - (n, g)-strategies for Prover are sufficient
  - A balancing strategy is an (n, g)-strategy
- Remarks
  - Bisimulation equivalence in the nondeterministic case
  - A complexity bound in the deterministic case

First-order regular terms; partial mapping $E : \mathbb{N}^* \to \mathcal{F} \cup \mathcal{V}$

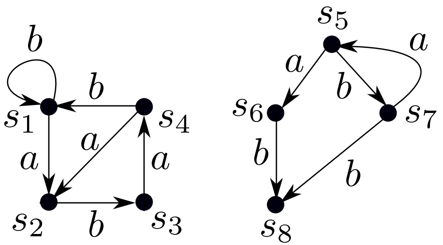


Substitutions $\sigma : \mathcal{V} \rightarrow \text{TERMS}_{\mathcal{F}}$; terms $E\sigma$, $E\sigma_1\sigma_2$, ...



Finite-support restriction: $\text{SUPP}(\sigma) = \{x_i \mid \sigma(x_i) \neq x_i\}$ is finite
Composing substitutions: $(\sigma_1\sigma_2)(x_i) = (\sigma_1(x_i))\sigma_2$
Associativity: $(E\sigma_1)\sigma_2 = E(\sigma_1\sigma_2)$; note that $x_i\sigma = \sigma(x_i)$

(Det-)labelled transition systems (LTSs); trace equivalence



$\mathcal{L} = (\mathcal{S}, Act, (\xrightarrow{a})_{a \in Act})$

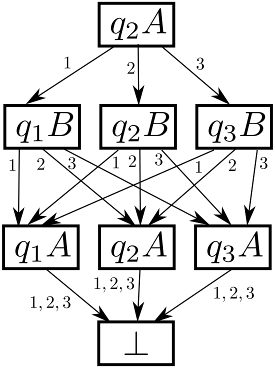
$\textsc{EqLv}(s_1, s_2) = 0$
$\textsc{EqLv}(s_1, s_5) = 2$
$\textsc{EqLv}(s_1, s_4) = \omega$

$\boxed{s \sim t}$ if $\forall w \in Act^* : s \xrightarrow{w} \Leftrightarrow t \xrightarrow{w}$; $\boxed{s \sim_k t}$ if
$\forall w \in Act^{\leq k} : s \xrightarrow{w} \Leftrightarrow t \xrightarrow{w}$

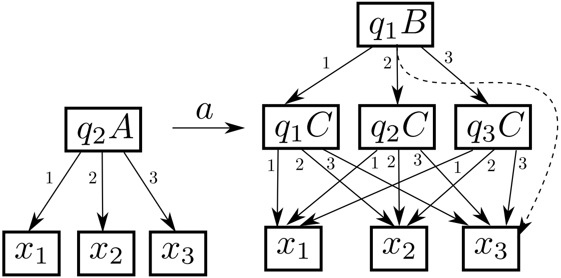
- $\mathcal{S} \times \mathcal{S} = \sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \cdots$. $\cap_{k \in \mathbb{N}} \sim_k = \sim$.
- If $\textsc{EqLv}(s, t) = k$ and $\textsc{EqLv}(s, s') \geq k+1$ then $\textsc{EqLv}(s', t) = k$ (replacing a pair-element with a "more equivalent state" does not affect the eq-level of the pair)
- In any deterministic LTS eq-level drops by at most 1 in one step:
  - If $s \xrightarrow{a} s'$, $t \xrightarrow{a} t'$ then $\textsc{EqLv}(s', t') \geq \textsc{EqLv}(s, t) - 1$.
  - If $\textsc{EqLv}(s, t) = k < \omega$ then there is a such that $s \xrightarrow{a} s'$, $t \xrightarrow{a} t'$, $\textsc{EqLv}(s', t') = \textsc{EqLv}(s, t) - 1$.

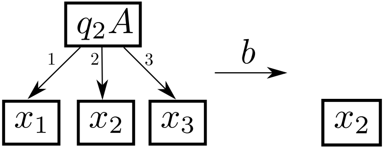
(D)pda from a first-order term perspective

$Q = \{q_1, q_2, q_3\}$
configuration $q_2 ABA$



(pushing) rule $q_2A \xrightarrow{a} q_1BC$





(popping) rule $q_2A \xrightarrow{b} q_2$

(Det-) first-order grammars as generators of (det-)LTSs

A det-first-order grammar

$\mathcal{G} = (\mathcal{N}, Act, \mathcal{R}) = (\{A, B\}, \{a, b\}, \{r_1, r_2, r_3, r_4\})$

$r_1 : Ax_1 \xrightarrow{a} ABx_1$

$r_2 : Ax_1 \xrightarrow{b} x_1$

$r_3 : Bx_1 \xrightarrow{a} BAx_1$

$r_4 : Bx_1 \xrightarrow{b} x_1$

Generally rules of $\mathcal{G} = (\mathcal{N}, Act, \mathcal{R})$ are of the form

$$r : Yx_1x_2 \dots x_m \xrightarrow{a} E$$

where $Y \in \mathcal{N}$, arity$(Y) = m$, $a \in Act$, and E is a finite term over $\mathcal{N}$ in which each occurring variable is from the set $\{x_1, x_2, \dots, x_m\}$.

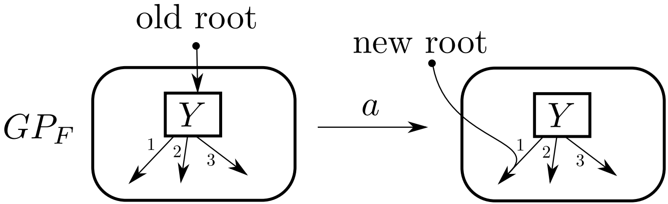
$\mathcal{G}$ is deterministic if there is at most one rule for each (Y, a).

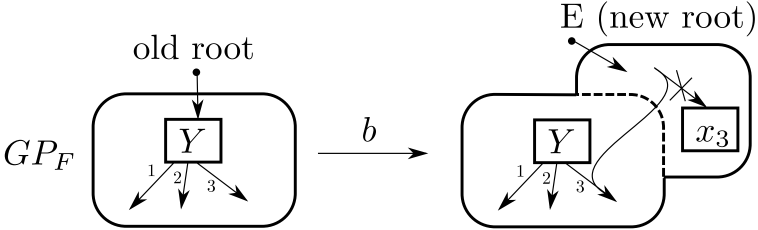
$(Yx_1 \dots x_m)\sigma \xrightarrow{a} E\sigma$ ... in the LTS $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}} = (\textsc{Terms}_{\mathcal{N}}, Act', (\xrightarrow{a})_{a \in Act'})$

Convention. In $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$ we have $x_i \not\sim_1 H$ if $H \neq x_i$ (e.g. if $H = x_j$ for $j \neq i$).

Applying the rules (to GP_F, for a regular F)

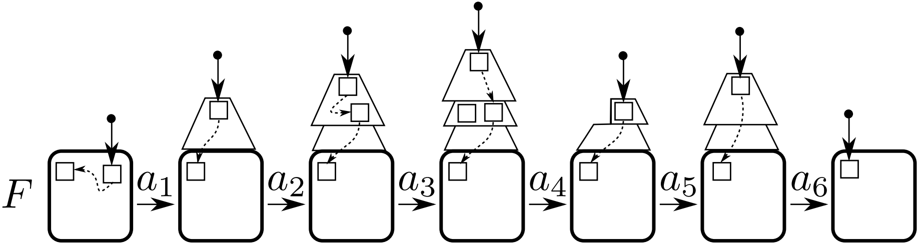
$$Yx_1x_2x_3 \xrightarrow{a} x_1$$





$$Yx_1x_2x_3 \xrightarrow{b} E$$

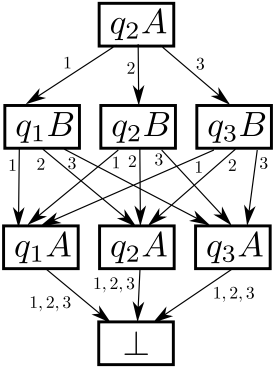
A path in $\mathcal{L}_{\mathcal{G}}^{\mathrm{A}}$

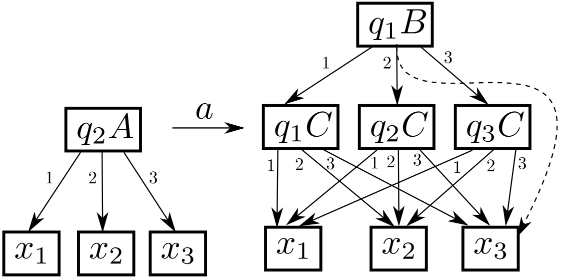


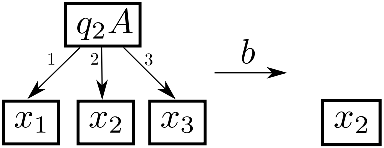
(D)pda from a first-order term perspective

$Q = \{q_1, q_2, q_3\}$
configuration q_2ABA

(pushing) rule $q_2A \xrightarrow{a} q_1BC$







(popping) rule $q_2A \xrightarrow{b} q_2$

Semidecidability of the nonequivalence

Problem $\textsc{Trace-Eq-Det-G}$:

Input: a det-first-order grammar $\mathcal{G} = (\mathcal{N}, Act, \mathcal{R})$, and
(graph presentations of) two input terms T_{in}, U_{in}.
Question: is $T_{in} \sim U_{in}$ in $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$?

Lemma

There is an algorithm with the following property:
it (halts and) computes $\textsc{EqLv}(T_{in}, U_{in})$ *for an instance* $\mathcal{G}, T_{in}, U_{in}$ *of* $\textsc{Trace-Eq-Det-G}$ *iff* $T_{in} \not\sim U_{in}$ *in* $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$.
Thus the complement of $\textsc{Trace-Eq-Det-G}$ *is semidecidable.*

Simple properties of $\sim_k$ and $\sim$

For two substitutions $\sigma, \sigma' : \mathcal{V} \to \textsc{Terms}_{\mathcal{N}}$ we define

$$\boxed{\sigma \sim_k \sigma'} \text{ if } \sigma(x_i) \sim_k \sigma'(x_i) \text{ for all } x_i \in \mathcal{V}.$$

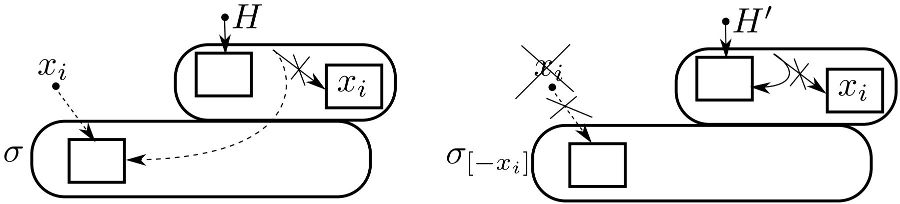
Proposition

(1) If $E \sim_k F$ then $E\sigma \sim_k F\sigma$. Hence $\textsc{EqLv}(E,F) \leq \textsc{EqLv}(E\sigma, F\sigma)$.
(2) If $\sigma \sim_k \sigma'$ then $E\sigma \sim_k E\sigma'$. Hence $\textsc{EqLv}(\sigma,\sigma') \leq \textsc{EqLv}(E\sigma, E\sigma')$.

Proposition

If $\textsc{EqLv}(E,F) = k < \ell = \textsc{EqLv}(E\sigma, F\sigma)$ (where $\ell \in \mathbb{N} \cup \{\omega\}$) then there are some $x_i \in \textsc{supp}(\sigma)$, $H \neq x_i$, and a word w, $|w| = k$, such that $E \xrightarrow{w} x_i$, $F \xrightarrow{w} H$ or $E \xrightarrow{w} H$, $F \xrightarrow{w} x_i$; moreover, $\sigma(x_i) \sim_{\ell-k} H\sigma$.

A limit replacement (of $\sigma(x_i)$ with $H\sigma$, where $H \neq x_i$)



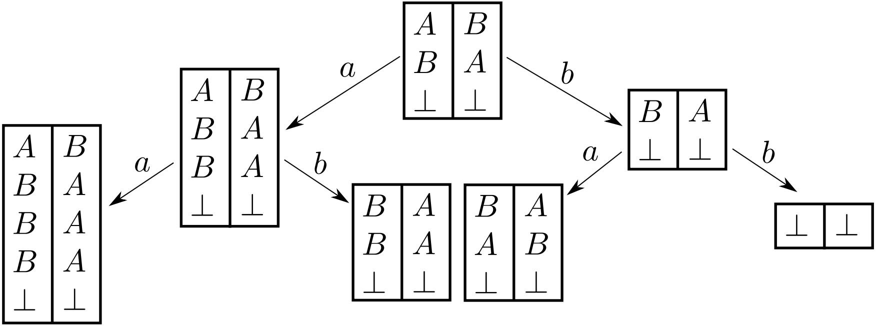
Proposition

$\mathrm{EQLV}(H\sigma, H'\sigma) > \mathrm{EQLV}(\sigma(x_i), H'\sigma)$, *or*
$\mathrm{EQLV}(H\sigma, H'\sigma) = \mathrm{EQLV}(\sigma(x_i), H'\sigma) = \omega$.
Hence $\sigma(x_i) \sim_k H\sigma$ *implies* $\sigma(x_i) \sim_k H'\sigma = H'\sigma_{[-x_i]}$.

k-distance regions $\textsc{Reg}(T, U, k)$

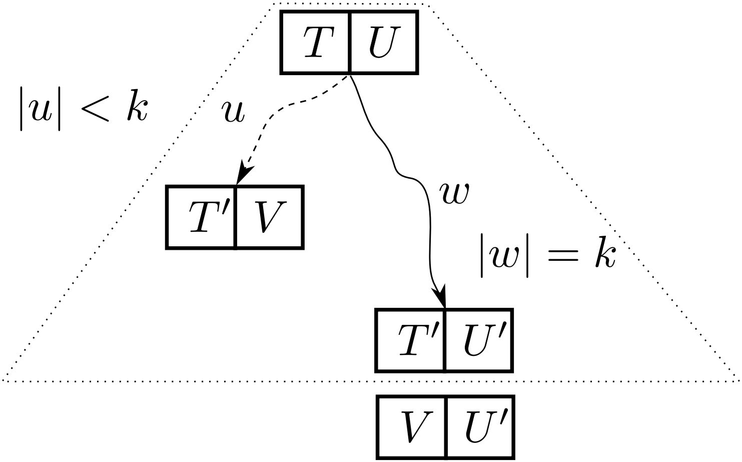
$\mathcal{G}$: $\quad Ax_1 \xrightarrow{a} ABx_1,\ Ax_1 \xrightarrow{b} x_1,\ Bx_1 \xrightarrow{a} BAx_1,\ Bx_1 \xrightarrow{b} x_1$

The 2-distance region $\textsc{Reg}(T, U, 2)$ for $(T, U) = (AB\bot, BA\bot)$



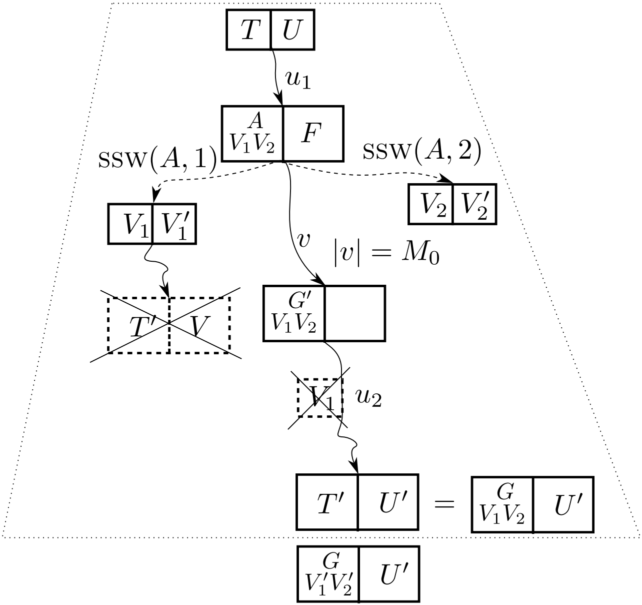
If $T \not\sim U$, $T \sim_k U$ then any least eq-level pair in $\textsc{Reg}(T, U, k)$ is at the bottom, i.e. in $\textsc{Reg}(T, U, k) \setminus \textsc{Reg}(T, U, k{-}1)$.

Case 1 of left-balancing



(T', U') is a least eq-level pair $\implies$ $\textsc{EqLv}(V, U') = \textsc{EqLv}(T', U')$.

Case 2 of left-balancing



$V_1 \sim_{\ell+1} V_1'$,
$V_2 \sim_{\ell+1} V_2'$,

$\sigma(x_1) = V_1$,
$\sigma(x_2) = V_2$,

$\sigma'(x_1) = V_1'$,
$\sigma'(x_2) = V_2'$,

$\sigma \sim_{\ell+1} \sigma'$
$G\sigma \sim_{\ell+1} G\sigma'$

$$\boxed{\mathrm{EqLv}(T', U') = \ell}$$

$(T', U') = (G\sigma, U')$

$\mathrm{EqLv}(G\sigma', U') = \ell$

Prover-Refuter game

1. A det-first-order grammar $\mathcal{G} = (\mathcal{N}, Act, \mathcal{R})$ is given.
2. Prover produces ("guesses") a finite set $\textsc{Basis}$ of pairs of terms.
3. An input pair (T_{in}, U_{in}) is given.
4. Refuter chooses $(T_0, U_0) \in \{(T_{in}, U_{in})\} \cup \textsc{Basis}$, claiming that (T_0, U_0) has the least eq-level.
5. For $i = 0, 1, 2, \ldots,$ Phase i is performed, i.e.:
    1. Prover chooses $k > 0$, and $\textsc{Reg}(T_i, U_i, k)$ is constructed; if $T_i \not\sim_k U_i$ then Prover loses (the play ends).
    2. Refuter chooses $(T'_i, U'_i) \in \textsc{Reg}(T_i, U_i, k) \smallsetminus \textsc{Reg}(T_i, U_i, k{-}1)$ and w_i, $|w_i| = k$, such that $T_i \xrightarrow{w_i} T'_i$, $U_i \xrightarrow{w_i} U'_i$; if not possible, Prover wins. Refuter claims that (T'_i, U'_i) has the least eq-level in $\textsc{Reg}(T_i, U_i, k)$.
    3. Prover produces (T_{i+1}, U_{i+1}) from (T'_i, U'_i) as follows:
        - either she puts $T_{i+1} = T'_i$, $U_{i+1} = U'_i$ (no-change),
        - or she makes a left-balancing step, or a right-balancing step.

       ($\textsc{EqLv}(T_{i+1}, U_{i+1}) = \textsc{EqLv}(T'_i, U'_i)$ if Refuter's claim in 5.b is true.)
    4. Prover either contradicts Refuter's claims by presenting a proof (a finite algorithmically verifiable sequence of deductions), in which case Prover wins, or lets the play proceed with Phase $i{+}1$.

Example of a play of the Prover-Refuter game

1. $\mathcal{G}$: $Ax_1 \xrightarrow{a} ABx_1,\ Ax_1 \xrightarrow{b} x_1,\ Bx_1 \xrightarrow{a} BAx_1,\ Bx_1 \xrightarrow{b} x_1$ is given.
2. Prover puts (guesses) $\mathrm{BASIS} = \{(x_1, x_1), (Ax_1, Bx_1)\}$.
3. $(T_{in}, U_{in}) = (AB\bot, BA\bot)$ is given.
4. Refuter chooses $(T_0, U_0) = (T_{in}, U_{in}) = (AB\bot, BA\bot)$.

5a. Prover chooses $k = 2$ and constructs $\mathrm{REG}(T_0, U_0, k)$.

5b. Refuter chooses $(T_0', U_0') = (ABBB\bot, BAAA\bot)$, with $w_0 = aa$.

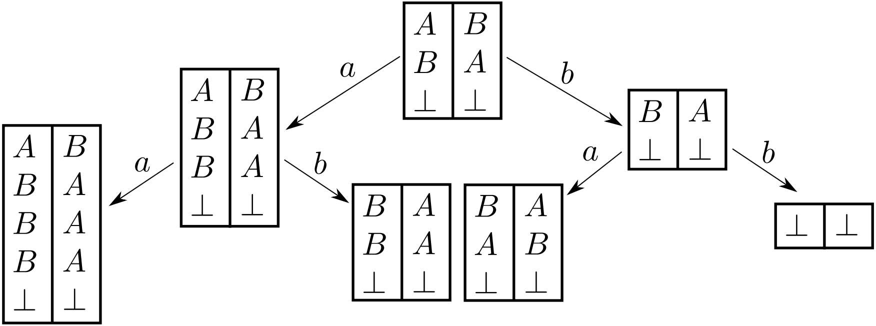
5c. Prover performs a left-balancing: using $(B\bot, A\bot)$ she puts $(T_1, U_1) = (ABBA\bot, BAAA\bot)$.

5d. Prover derives a contradiction by assuming Refuter's claims are true as follows, denoting $\mathrm{EQLV}(T_1, U_1) = \mathrm{EQLV}(T_0', U_0') = \ell$: since the eq-levels of $(A\bot, B\bot)$, $(AB\bot, BA\bot)$, $(ABB\bot, BAA\bot)$ are greater than ℓ, the eq-level of each of the following pairs (arising from (T_1, U_1) by successive subterm replacements) must be ℓ: $(ABAB\bot, BAAA\bot)$ $(ABAB\bot, BAAB\bot)$, $(ABAB\bot, BABA\bot)$, $(ABAB\bot, BABB\bot)$, $(ABAB\bot, BBAA\bot)$, $(ABAB\bot, BBAB\bot)$. The last pair is an instance of a basis-pair, since $(Ax_1, Bx_1) \in \mathrm{BASIS}$.

k-distance regions $\textsc{Reg}(T, U, k)$

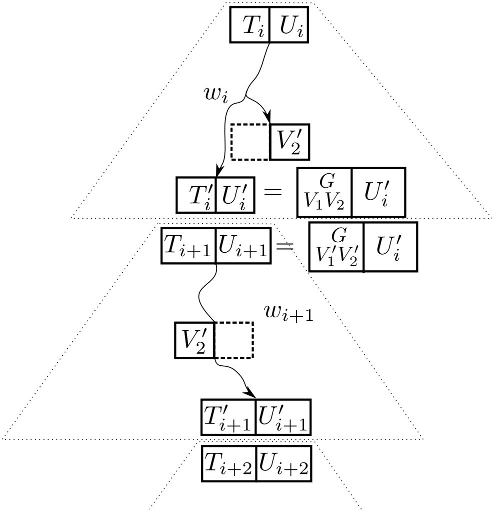
$\mathcal{G}$: $Ax_1 \xrightarrow{a} ABx_1$, $Ax_1 \xrightarrow{b} x_1$, $Bx_1 \xrightarrow{a} BAx_1$, $Bx_1 \xrightarrow{b} x_1$

The 2-distance region $\textsc{Reg}(T, U, 2)$ for $(T, U) = (AB\bot, BA\bot)$



If $T \not\sim U$, $T \sim_k U$ then any least eq-level pair in $\textsc{Reg}(T, U, k)$ is at the bottom, i.e. in $\textsc{Reg}(T, U, k) \smallsetminus \textsc{Reg}(T, U, k{-}1)$.

A left balancing phase i followed by a no-change phase $i+1$



T_i U_i
w_i
V_2'
T_i' U_i' $=$ $\overset{G}{V_1V_2}$ U_i'
T_{i+1} U_{i+1} $=$ $\overset{G}{V_1'V_2'}$ U_i'
w_{i+1}
V_2'
T_{i+1}' U_{i+1}'
T_{i+2} U_{i+2}

(T_0, U_0)

(T_1, U_1)

(T_2, U_2)

(T_3, U_3)

...

eq-level decreasing if Refuter's claims are true

Soundness of the P-R game; completeness remains

Lemma

There is an algorithm with the following property:
given a det-first order grammar $\mathcal{G}$ *and* T_{in}, U_{in},

the algorithm halts and produces some $\textsc{Basis}$
with which Prover can force her win for $\mathcal{G}$, T_{in}, U_{in}

iff

there is such $\textsc{Basis}$*;*
in this case $T \sim U$ *for all* $(T, U) \in \{(T_{in}, U_{in})\} \cup \textsc{Basis}$.

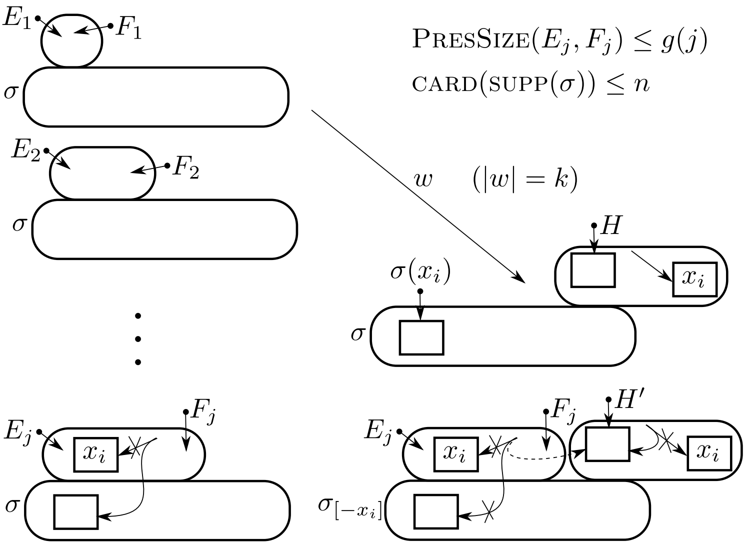
After showing completeness (for any $\mathcal{G}$ there is a sufficient $\textsc{Basis}$), we get:

Theorem

Trace equivalence for det-first-order grammars (i.e., the problem $\textsc{Trace-Eq-Det-G}$*) is decidable.*

An (n, g)-strategy for $\mathcal{G}$ implies a sufficient basis

$\textsc{Basis} = \{(E, F) \mid E \sim F, \textsc{PresSize}(E, F) \leq \mathcal{B}\}$ for large $\mathcal{B} \in \mathbb{N}$

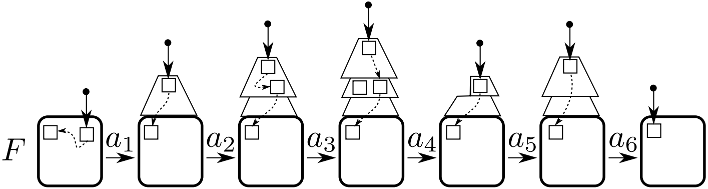


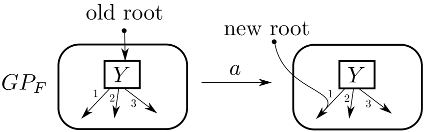
$\textsc{PresSize}(E_j, F_j) \leq g(j)$

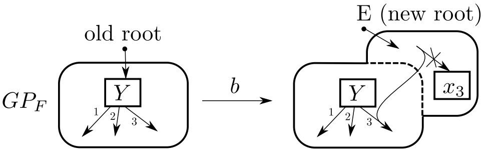
$\textsc{card}(\textsc{supp}(\sigma)) \leq n$

(Y,j)-sink-words; shortest sink-words $\textsc{ssw}(Y,j)$

$w \in Act^*$ is a (Y,j)-sink-word, $1 \le j \le m = \text{arity}(Y)$, if $Yx_1 \dots x_m \xrightarrow{w} x_j$







$M_0 = 1 + \max\{\, |\textsc{ssw}(Y,j)| \mid Y \in \mathcal{N}, 1 \le j \le \text{arity}(Y)\}$

Restricted balancing (in Phase i); balancing strategy

$M_1 = M_0 \cdot (2+(2M_0{-}1)\cdot \textsc{StepDepthInc})$ … Prover puts $k = M_1$ in 5a

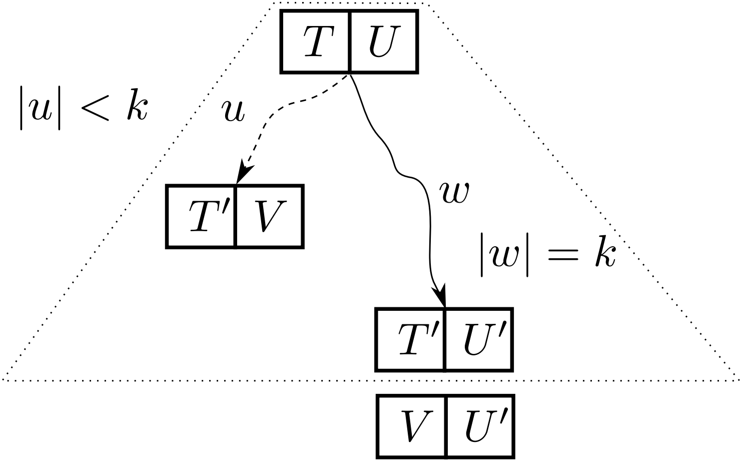
(Restricted) left balancing:

1. If there is some (T'_i, V) in $\textsc{Reg}(T_i, U_i, M_1{-}1)$, Prover (chooses one such pair and) puts $T_{i+1} = V$, $U_{i+1} = U'_i$. (Case 1 of left balancing)
2. Otherwise: Case 2 of left balancing for the last root-performable (sub)path of length M_0 in $T_i \xrightarrow{w_i} T'_i$ (if there is some) ….
3. If none of 1, 2 applies, hence $T_i \xrightarrow{w_i} T'_i$ is "M_0-sinking", then no left-balancing is possible.

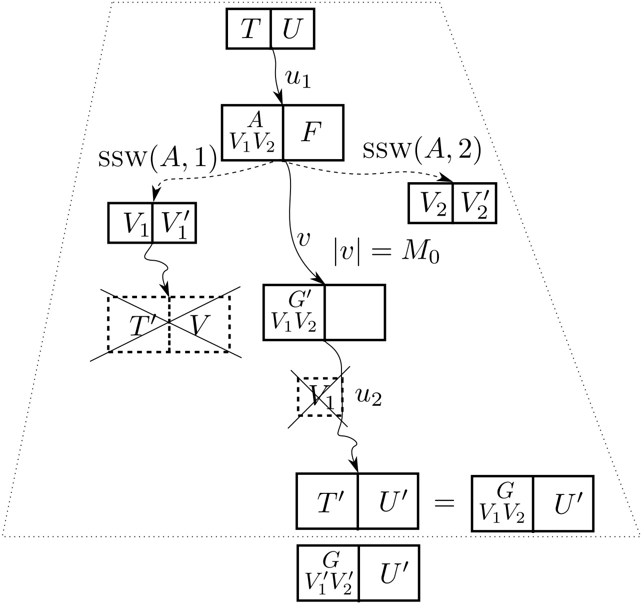
In 1,2, U_i … the balancing pivot, (T_{i+1}, U_{i+1}) … the balancing result.

Additional restriction:
Prover must balance when possible but she cannot do a left (right) balancing in Phase i if a right (left) balancing was done in Phase $i{-}1$.
(Each side-switch needs a separating no-change phase.)

Case 1 of left-balancing



Case 2 of left-balancing



A left balancing phase i followed by a no-change phase $i+1$

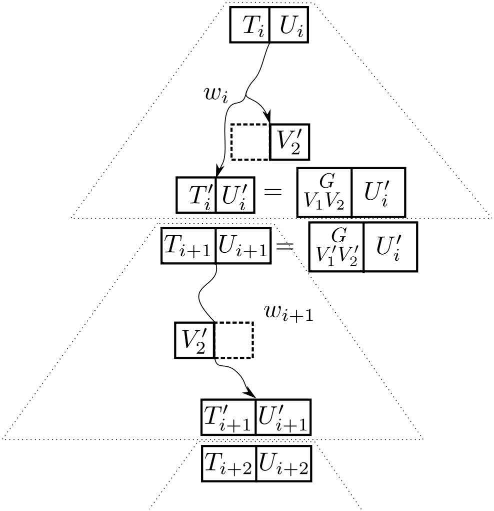
U_i

$\downarrow u$

V_2'

$\downarrow u'$

T_{i+2}



T_i U_i
w_i
V_2'
T_i' U_i' = G $V_1 V_2$ U_i'
T_{i+1} U_{i+1} = G $V_1' V_2'$ U_i'
w_{i+1}
V_2'
T_{i+1}' U_{i+1}'
T_{i+2} U_{i+2}

U_i

$\downarrow w_i$

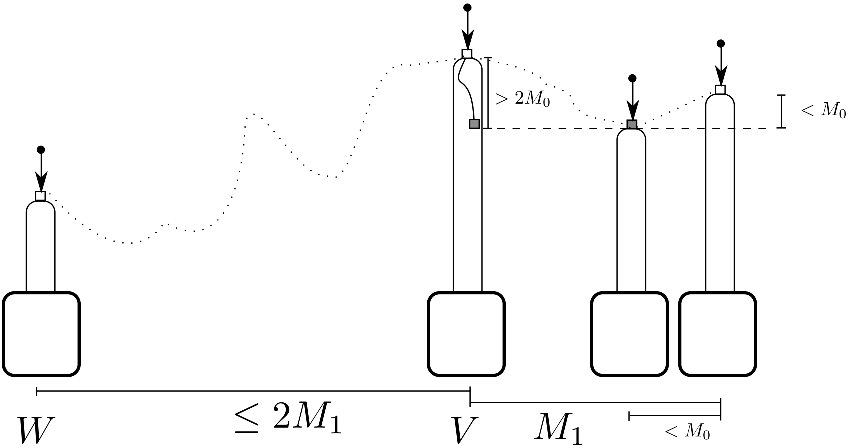
U_{i+1}

$\downarrow w_{i+2}$

U_{i+2}

Balancing pivots are on a special path in $\mathcal{L}^{\mathrm{A}}_{\mathcal{G}}$

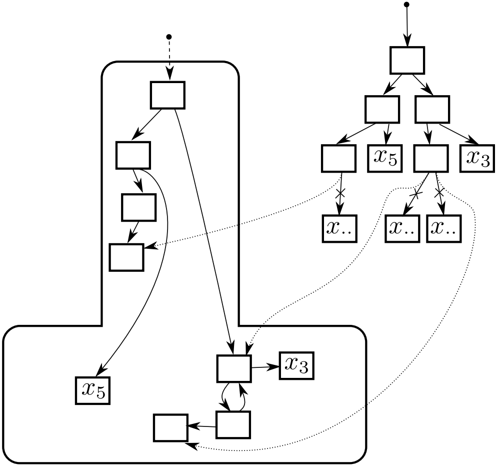
$W_1 \xrightarrow{v_1} W_2 \xrightarrow{v_2} W_3 \xrightarrow{v_3} \cdots$



Recall $M_1 = M_0 \cdot (2 + (2M_0 - 1) \cdot \textsc{StepDepthInc})$

Presenting V as $V = (\textsc{Top}_d^V)\sigma$

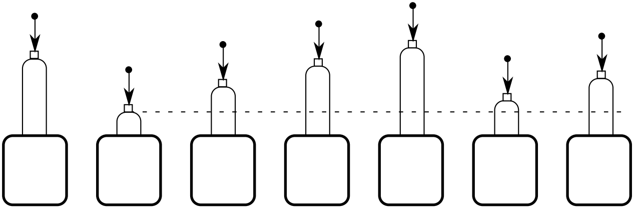
$\textsc{supp}(\sigma) \subseteq \{x_1, x_2, \ldots, x_{c^d}\}$ where $c = \max\{\,\text{arity}(Y) \mid Y \in \mathcal{N}\,\}$



$d = 3$ in the example

Balancing strategy is an (n, g)-strategy (for the given $\mathcal{G}$)

If the pivot path $W_1 \xrightarrow{v_1} W_2 \xrightarrow{v_2} W_3 \xrightarrow{v_3} \cdots$ is finite or visits some V' infinitely often then we have a repeat ($(T_j, U_j) = (T_i, U_i)$ for $j > i$). Otherwise we have a "stair-base":



$$W_1 \xrightarrow{u} V = (Yx_1 \dots x_m)\sigma' \xrightarrow{u'} H_1\sigma' \xrightarrow{v_{k+1}} H_2\sigma' \xrightarrow{v_{k+2}} \cdots$$

$$\text{where } (Yx_1 \dots x_m) \xrightarrow{u'} H_1 \xrightarrow{v_{k+1}} H_2 \xrightarrow{v_{k+2}} \cdots,$$

and $H_j\sigma' = W_{k+j}$ $(j = 1, 2, \dots)$ are the pivots after $V = (Yx_1 \dots x_m)\sigma'$. Putting $V = (\mathrm{TOP}^{V}_{M_1})\sigma = ((Yx_1 \dots x_m)\sigma'')\sigma$, we have $W_{k+j} = H_j\sigma''\sigma$, and the bal-result with W_{k+j} is $(E_j\sigma, F_j\sigma)$, where E_j, F_j are finite terms; we can easily find function $g : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{PRESSIZE}(E_j, F_j) \leq g(j)$.

Plán

- poznámky k předmětům (mgr. studium)
  - 460-4005/01 - Teoretická informatika (TI)
  - 460-4043/01 - Vybrané partie teoretické informatiky (VPTI)
  - 460-4016/01 - Modelování a verifikace (MaV)
  - 460-4006 - Petriho sítě I (PES I); 460-4019 - Petriho sítě II (PES II)
  - 460-4037/01 - Teorie her (TEH)
- výzkumná práce v oblasti (mezí automatizované) verifikace
  - aktuální výsledek:
    P. Jančar:
    Decidability of DPDA Language Equivalence via First-Order Grammars
    (nový důkaz ekvivalence deterministických zásobníkových automatů)
    (přijato na 27th Annual ACM/IEEE Symposium on
    Logic in Computer Science (LICS 2012), June 2012)