Counter Systems for Data Logics

Stéphane Demri

Laboratoire Spécification et Vérification (LSV)
ENS de Cachan & CNRS & INRIA

12th European Conference on Logics in Artificial Intelligence
September 13–15, 2010, Helsinki, Finland

Models with Data

Ubiquity of data words
[Bouyer, IPL 02]

- Data word

$$\begin{matrix} a_1 & a_2 & a_3 & \cdots \\ d_1 & d_2 & d_3 & \cdots \end{matrix}$$

  - Each a_i belongs to a finite alphabet Σ.
  - Each d_i belongs to an infinite domain D.

- Timed word [Alur & Dill, TCS 94]

$$\begin{matrix} a & b & c & a & a & b \\ 0 & 0.3 & 1 & 2.3 & 3.5 & 3.51 \end{matrix}$$

- Runs from counter systems

$$\begin{matrix} q_0 & q_2 & q_3 & q_2 & q_3 & q_2 \\ 0 & 0 & 1 & 2 & 3 & 4 \end{matrix}$$

- Integer arrays [Habermehl & Iosif & Vojnar, FOSSACS'08]

$$t[0] \quad t[1] \quad t[2] \quad t[3] \quad t[4] \quad t[5] \ldots$$

Finite alphabet and infinite domain

a	a	b	d	a	b
URL_1	URL_2	URL_1	URL_2	URL_3	URL_3

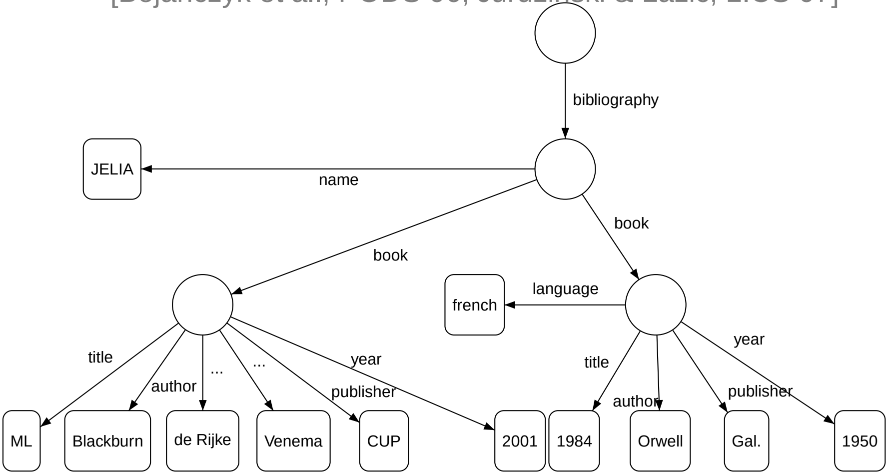
a	a	b	d	a	b
3	2.5	3	2.5	4	4

$a \quad a \quad b \quad d \quad a \quad b$

Data trees

Extension to data trees (XML documents with values).

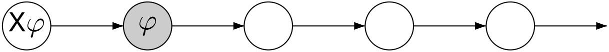
[Bojańczyk et al., PODS 06; Jurdziński & Lazić, LICS 07]



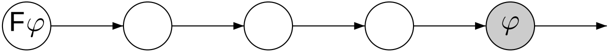
Formalisms for Data Words – Temporal Logics

Linear-time temporal operators

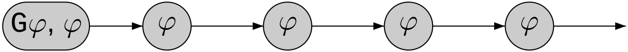
Xφ: next-time φ



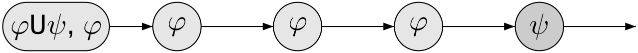
Fφ: sometimes φ



Gφ: always φ



φUψ: φ until ψ



A mechanism for handling data

- Case analyses in formulae are not sufficient with infinite domains.
- A register can store a data value and equality tests are performed between registers and current data values.
- Storing a value in a register:

$$\downarrow_r \varphi \stackrel{\text{def}}{=} \exists \mathsf{y}_r \ (\mathsf{y}_r = \mathsf{x}) \wedge \varphi$$

- Equality test between a register and a value: $\uparrow_r \stackrel{\text{def}}{=} \mathsf{y}_r = \mathsf{x}$. (in this talk, restriction to the simple equality tests)
- All data values at distinct positions are distinct:

$$\mathtt{G}(\downarrow_r \mathtt{XG}\neg \uparrow_r)$$

- Generalization with memory logics, e.g. memory bags have operations “register”, “forget” and “erase”.

[Mera, PhD thesis 09]

Freeze operator

- Freeze quantifier in hybrid logics.
  [Goranko 94; Blackburn & Seligman, JOLLI 95]

- Temporal semantics of imperative programs.
  [Manna & Pnueli, 1992]
  Program variable x never decreases below its initial value:

$$\exists y\ (x = y)\ \wedge\ \mathtt{G}(x \geq y)$$

- Freeze quantifier in real-time logics.
  [Alur & Henzinger, JACM 94]
  $y \cdot \varphi(y)$ binds the variable y to the current time t.

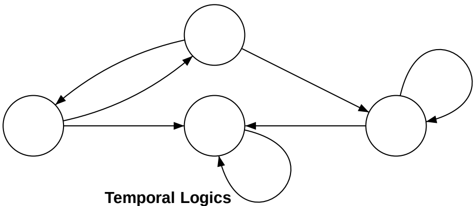
- Predicate λ-abstraction [Fitting, JLC 02].
  $\langle y \cdot \mathrm{F}\, P(y)\rangle(c)$: current value of constant c satisfies the predicate P.

- See also description logics over concrete domains.
  [Baader & Hanschke, IJCAI'91; Lutz, TOCL 04]

Hybrid logics as data logics

- Most standard models for modal logics are graphs in which nodes are labelled by propositional valuations.
- For a given formula, the set of propositional valuations is a finite alphabet.
- $\downarrow_y \varphi$: φ holds true in the variant model where proposition y is true only at the current state.
  [Goranko 94; Blackburn & Seligman, JOLLI 95].
- Models are enriched with node adresses.
- "Every reachable state can be visited infinitely often":

$AG \downarrow_y E\, XF\, y$



LTL with registers: $\mathrm{LTL}^{\downarrow}$

- $\mathrm{LTL}^{\downarrow}$ formulae:

$$\varphi \quad ::= \quad a \mid \uparrow_r \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \mathsf{U} \varphi \mid \mathsf{X}\varphi \mid \downarrow_r \varphi$$

  where $a \in \Sigma$ and $r \in \mathbb{N}^{+}$.

- Register valuation f: finite partial map from $\mathbb{N}^{+}$ to $\mathbb{N}$ $(= D)$.

- Models: finite or infinite data words over the alphabet Σ.

- Satisfaction relation:

$$\sigma, i \models_f \uparrow_r \quad \stackrel{\text{def}}{\Leftrightarrow} \quad r \in \mathrm{dom}(f) \text{ and } f(r) = d_i$$

$$\sigma, i \models_f \downarrow_r \varphi \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \sigma, i \models_{f[r \mapsto d_i]} \varphi$$

  (d_i: data value at position i)

- Unlike standard LTL, $\mathrm{LTL}^{\downarrow}$ can store a data value and perform equality tests.

Examples

- Nonce property: $\mathtt{G}(\downarrow_1 \mathtt{XG}\neg \uparrow_1)$.

$$\downarrow_1 \mathsf{X} \uparrow_1 \approx \mathsf{x} = \mathbf{X}\mathsf{x}$$

$$a \;\; a \;\; b \;\; d \;\; a \;\; b \;\; , 0 \not\models \mathsf{F}(a \wedge \downarrow_1 \mathsf{XF}(a \wedge \uparrow_1))$$

An another view on $\mathrm{LTL}^{\downarrow}$

- Standard LTL models are of the form $\mathbb{N} \rightarrow \mathcal{P}(\mathrm{PROP})$ for some countably infinite set PROP of atomic propositions.

- An LTL formula φ built over $\{\mathrm{p}_1, \ldots, \mathrm{p}_k\}$ constrains the models only for $\{\mathrm{p}_1, \ldots, \mathrm{p}_k\}$

- No LTL formula characterizes the class of models for which any two distinct positions have distinct valuations.

- $\mathrm{LTL}^{\downarrow}$ = extension of LTL (with standard models) where the registers store valuations in $\mathcal{P}(\mathrm{PROP} \setminus \mathrm{PROP}_k)$ and the alphabet is $\mathcal{P}(\mathrm{PROP}_k)$ with $\mathrm{PROP}_k = \{\mathrm{p}_1, \ldots, \mathrm{p}_k\}$.

Complexity of satisfiability problems

- Finitary and infinitary satisfiability problem for LTL are PSPACE-complete. [Sistla & Clarke, JACM 85]

- What about $\mathrm{LTL}^{\downarrow}$ with one register, with all registers etc.?

- Infinitary satisfiability problem for $\mathrm{LTL}^{\downarrow}$ restricted to X and F and to a single register is undecidable.

- Finitary satisfiability problem for $\mathrm{LTL}^{\downarrow}$ restricted to a single register is decidable but nonprimitive recursive.

  [Demri & Lazić, TOCL 09]

- Finitary satisfiability problem for $\mathrm{LTL}^{\downarrow}$ restricted to F and
  - to a single register is nonprimitive recursive too.
  - to two registers is undecidable.

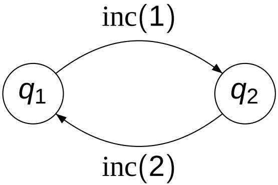
  [Figueira & Segoufin, MFCS'09]

- Nonprimitive recursiveness uses [Schnoebelen, IPL 02].

How Counter Systems Enter into the Play

Counter automata (CA)

- Counter system = finite-state automaton + counters.
- Counter: program variable interpreted by a non-negative integer.



- Counter automaton $\mathcal{S} = (Q, n, \delta)$
  - Finite set of control states Q.
  - Transitions in $\delta \subseteq Q \times \{\mathrm{zero}(i), \mathrm{inc}(i), \mathrm{dec}(i) : i \in [1, n]\} \times Q$.
  - Dimension n (number of counters).
- Runs of the form

$$\rho = \begin{matrix} q_0 \\ \vec{x}_0 \end{matrix} \rightarrow \begin{matrix} q_1 \ (\in Q) \\ \vec{x}_1 \ (\in \mathbb{N}^n) \end{matrix} \rightarrow \begin{matrix} q_2 \\ \vec{x}_2 \end{matrix} \rightarrow \ldots$$

Reachability problems

- Reachability problem:

  Input: counter automaton $\mathcal{S}$, $(q, \vec{0})$ and $(q', \vec{0})$.

  Question: is $(q, \vec{0}) \xrightarrow{*} (q', \vec{0})$?

- Control state reachability problem:

  Input: counter automaton $\mathcal{S}$, $(q, \vec{0})$ and q'.

  Question: is $(q, \vec{0}) \xrightarrow{*} (q', \vec{x'})$ for some $\vec{x'}$?

- Control state repeated reachability problem:

  Input: counter automaton $\mathcal{S}$, $(q, \vec{0})$ and q_f.

  Question: is there an infinite run from $(q, \vec{x})$ such that q_f is repeated infinitely often?

- Covering problem (extending control state reachability):

  Input: counter automaton $\mathcal{S}$, $(q, \vec{0})$ and $(q', \vec{x'})$.

  Question: is $(q, \vec{0}) \xrightarrow{*} (q', \vec{x''})$ with $\vec{x'} \preceq \vec{x''}$?

  ($\preceq$ is defined pointwise)

Counter automata generate data words

- A counter automaton and an initial configuration generate a set of runs viewed as data words with multiple data values.
- The finite alphabet is Q.
- Extension of freeze operators to $\downarrow_r^j$ and $\uparrow_r^j$ with $j \in [1, n]$.

Turing-completeness of Minsky machines

- A counter stores a single natural number.
- A Minsky machine can be viewed as a deterministic finite-state automaton with two counters.
- Operations on counters:
  - Check whether the counter is zero.
  - Increment the counter by one.
  - Decrement the counter by one if nonzero.
- Halting problem ($\approx$ control state reachability problem):

  input: a Minsky machine M;
  question: is the unique computation halts?
- The halting problem is undecidable and Minsky machines are Turing-complete. [Minsky, 67]

Reachability Problems for Gainy CA

Gainy counter automata

- Faulty systems perform errors such as losses or gains, e.g., see works on lossy channel systems.
  [Abdulla & Jonsson, IC 96]

- Three ways to model gainy counter automata:
  1. Standard CA (Q, n, δ) such that for $q \in Q$ and $i \in [1, n]$, $q \xrightarrow{\mathrm{inc}(i)} q \in \delta$.
  2. Alternative one-step relation: $(q, \vec{x}) \xrightarrow{t}_g (q', \vec{x'})$ iff there are $\vec{y}, \vec{y'}$ in $\mathbb{N}^n$ such that

     $$\vec{x} \preceq \vec{y} \ \text{ and } \ (q, \vec{y}) \xrightarrow{t} (q', \vec{y'}) \ (\text{exact step}) \ \text{ and } \ \vec{y'} \preceq \vec{x'}$$

  3. Gains occur in a lazy way: decrement on zero has no effect.

Benefits from Gainy CA

- Features:
  - Increment, decrement and zero-test.
  - Incrementation errors.

- Control state reachability problem is decidable but with a nonprimitive recursive complexity.
  See e.g., [Urquhart, JSL 99; Schnoebelen, IPL 02]

- Control state repeated reachability problem is undecidable.
  [Demri & Lazić, TOCL 09]
  (adapt a proof from [Ouaknine & Worrell, FOSSACS'06])

- These problems reduce to corresponding satisfiability problems for $\mathrm{LTL}^{\downarrow}$ restricted to X and F and to a single register.

Simulating Gainy CA

- Gainy CA $\mathcal{S}$ with initial configuration $(q_0, \vec{0})$.

- For $t \in \delta$, $\Sigma(t)$ denotes the instruction labelling it in $\Sigma = \{\mathrm{inc}(i), \mathrm{dec}(i), \mathrm{zero}(i) : i \in [1, n]\}$.

- Let us build φ in $\mathrm{LTL}^{\downarrow}$ s.t. φ is satisfiable iff $(\mathcal{S}, (q_0, \vec{0}))$ has an infinite run with q_f occurring infinitely often.

- φ is satisfiable only in models in which each position is labelled by a transition and by a value in $\mathbb{N}$.

- Infinite models of φ are of the form $(t_0, y_0), (t_1, y_1), (t_2, y_2), \cdots$ with $t_i \in \delta$ and $y_i \in \mathbb{N}$.

- For $I, J \in \mathbb{N}$, $I \sim J \overset{\text{def}}{\Leftrightarrow} y_I = y_J$.

Simulating gainy CA (II)

- Let us explain how the run from $(q_0, \vec{0})$ below is encoded.

$$(q_0, \vec{x_0}) \xrightarrow{a_0} (q_1, \vec{x_1}) \xrightarrow{a_1} \cdots \xrightarrow{a_{K-1}} (q_K, \vec{x_K}) \cdots$$

- Projection of the model over δ is

$$t_0 t_1 t_2 \cdots = q_0 \xrightarrow{a_0} q_1, q_1 \xrightarrow{a_1} q_2, \cdots$$

and q_f is repeated inifinitely often.

- Initial state is q_0:

$$\bigvee_{t = q_0 \xrightarrow{a} q} t$$

- The sequence of transitions respects δ:

$$\mathtt{G}\Big(\bigwedge_{t = q \xrightarrow{a} q' \in \delta} \big(t \Rightarrow \mathtt{X} \bigvee_{t' = q' \xrightarrow{a} q''} t'\big)\Big)$$

Simulating Gainy CA (III)

- Control state q_f is visited infinitely often: $\mathtt{GF} \bigvee_{t = q \xrightarrow{a} q_f} t$

- Each increment or decrement is associated to a unique value.

$$\mathrm{inc}(1) \quad \mathrm{inc}(1) \quad \mathrm{dec}(1) \quad \mathrm{inc}(1) \quad \mathrm{dec}(1) \quad \mathrm{dec}(1) \quad \mathrm{zero}(1)$$

- For $a \in \Sigma$, a is also used as a shortcut for $\bigvee_{t = q \xrightarrow{b} q' \in \delta,\ a = b} t$.

- For $i, j \in [1, n]$, there are no two positions for increments [resp. decrements] having the same value:

$$\mathtt{G}(\mathrm{inc}(i) \Rightarrow \neg(\downarrow_1 \mathtt{XF}(\uparrow_1 \wedge \mathrm{inc}(j)))) \quad \mathtt{G}(\mathrm{dec}(i) \Rightarrow \neg(\downarrow_1 \mathtt{XF}(\uparrow_1 \wedge \mathrm{dec}(j))))$$

Simulating Gainy CA (IV)

- The two next conditions are formulated in such a way to avoid using the until operator $\mathtt{U}$.

- For $i \in [1, n]$ and $J > I$, if $\Sigma(t_I) = \mathrm{inc}(i)$ and $\Sigma(t_J) = \mathrm{zero}(i)$, then there is no $K > J$ such that $\Sigma(t_K) = \mathrm{dec}(i)$ and $I \sim K$:

$$\mathtt{G}(\mathrm{inc}(i) \Rightarrow \downarrow_1 \neg(\mathsf{F}(\mathrm{zero}(i) \wedge (\mathsf{F}(\uparrow_1 \wedge \mathrm{dec}(i))))))$$

- For $i \in [1, n]$, if there are $J > I$ such that $\Sigma(t_I) = \mathrm{inc}(i)$ and $\Sigma(t_J) = \mathrm{zero}(i)$, then there is $K > I$ such that $\Sigma(t_K) = \mathrm{dec}(i)$ and $I \sim K$.

$$\mathtt{G}((\mathrm{inc}(i) \wedge \mathsf{F}\ \mathrm{zero}(i)) \Rightarrow \downarrow_1 (\mathsf{F}(\mathrm{dec}(i) \wedge \uparrow_1)))$$

- φ is satisfiable iff $(\mathcal{S}, (q_0, \vec{0}))$ has an infinite run such that q_f occurs infinitely often.

Gainy CA for $\mathrm{LTL}^{\downarrow}$ with one register !

- Control state repeated reachability problem for Gainy CA can be reduced to infinitary satisfiability for $\mathrm{LTL}^{\downarrow}$ restricted to one register. $\rightarrow$ undecidability

- Control state reachability problem for Gainy CA can be reduced to finitary satisfiability for $\mathrm{LTL}^{\downarrow}$ restricted to one register. $\rightarrow$ nonprimitive recursiveness

- In the finitary case, there is a converse reduction.

About nonprimitive recursiveness

- Control state reachability problem for gainy CA is nonprimitive recursive.
  See e.g., [Urquhart, JSL 99; Schnoebelen, MFCS'10]

- Ackermann function is nonprimitive recursive.
  (grows faster than any primitive recursive function)

- Decidable nonclassical logics with nonprimitive recursive complexity:
  - Products of modal logics with expanding domains by reduction from the reachability problem for lossy channel systems. [Gabelaia et al., APAL 06]
  - Relevance logic LR$+$ (and fragments) by introducing a branching extension of CA. [Urquhart, JSL 99]
  - Finitary Metric Temporal Logic MTL. [Ouaknine & Worrell, LICS'05]

Model-Checking Counter Automata

Motivations

- Model-checking with focus on data values

  1. To analyze runs of operational models with focus on data values (beyong control state reachability).
     E.g., “there is a value of counter 1 such that infinitely often counter 2 takes that value iff infinitely often counter 3 takes that value”:
     $$\mathtt{F} \downarrow_1^1 (\mathtt{GF} \uparrow_1^2 \Leftrightarrow \mathtt{GF} \uparrow_1^3)$$

  2. Model-checking rather than satisfiability.

- Current instance:

  - Operational models: classes of counter automata for which the reachability problem is decidable.

  - Most often, the reachability sets are definable in Presburger arithmetic (decidable first-order theory of $(\mathbb{N}, +)$) .

  - Specification language: LTL with registers.

Model-checking counter automata

- Infinitary model-checking problem $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow})$:

  Input: CA $\mathcal{S} = (Q, n, \delta)$, configuration $(q, \vec{x}) \in Q \times \mathbb{N}^n$, and a sentence $\varphi \in \mathrm{LTL}^{\downarrow}$ over alphabet Q;

  Question: Is there an infinite run ρ such that $\rho, 0 \models \varphi$?

- Undecidability for nondeterministic one-counter automata:

  - $\mathrm{MC}^{<\omega}(\mathrm{LTL}_1^{\downarrow}(\mathsf{X}, \mathsf{F}))$ is Σ_1^0-complete.

  - $\mathrm{MC}^{\omega}(\mathrm{LTL}_1^{\downarrow}(\mathsf{X}, \mathsf{F}))$ is Σ_1^1-complete.

- Reachability sets are semilinear but universal problem for one-counter automata with alphabet is undecidable.

  [Ibarra, MST 79]

Reversal-bounded counter automata

- Reversal: Alternation from nonincreasing mode to nondecreasing mode and vice-versa.

- Sequence with 3 reversals:

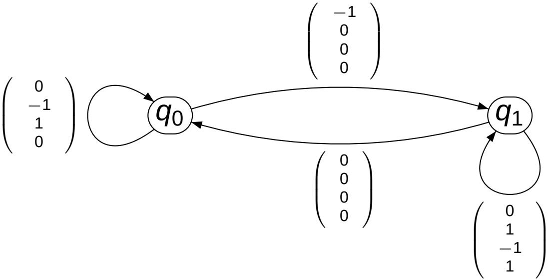
$$0011223334444\overline{3}33222\overline{3}33444455555\overline{4}$$

- Reversal-bounded counter automata: each run has a bounded number of reversals. [Ibarra, JACM 78]

- Reachability sets are effectively Presburger-definable. [Ibarra, JACM 78]

- Control state repeated reachability problem is decidable. [Dang & Ibarra & San Pietro, FST&TCS'01]

Vector addition systems with states (VASS)



- Succinct CA without zero-tests.
- Transitions of the form $q \xrightarrow{\vec{b}} q'$ with $\vec{b} \in \mathbb{Z}^n$, which is a shortcut for $\bigwedge_{i \in [1,n]} \mathsf{x}_i' = \mathsf{x}_i + \vec{b}(i)$.
- $\approx$ Petri nets (models of greater practical appeal).
- The reachability problem is decidable.

[Kosaraju, STOC'82; Mayr, SIAM 84]

Towards flatness

- $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow})$ restricted to reversal-bounded CA and to formulae with at most one register is undecidable.

- Flat formulae: positive occurrence of $\varphi_1 \mathsf{U} \varphi_2$ implies $\downarrow$ does not occur in φ_1.

- $\neg(q \ \mathsf{U} \downarrow_1^1 \varphi)$ is not a flat formula.

- Flatness is a standard means to regain decidability for memoryful linear-time temporal logics.

Introducing parameters

- $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow})$ restricted to reversal-bounded CA and to flat formulae is decidable. [Demri & Sangnier, FOSSACS'10]
- Decidability proof uses that the control state repeated reachability problem for parameterized reversal-bounded CA is decidable. [Ibarra et al., TCS 02]
- Transitions of the form $\mathit{add}(\mathrm{z})$ with parameter z.
- Reachability questions are relative to parameter values.

Formalisms for Data Words – First-Order Logics

First-order logic on data words

- Data word: nonempty finite sequence of pairs from $\Sigma \times \mathbb{N}$.

- Variable valuation *v* for a model σ: map from VAR to the positions of σ.

- Variables are interpreted as positions.

- Formulae of the logic $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ (Σ is a finite alphabet)

  $\varphi ::= a(\mathsf{x}) \mid \mathsf{x} \sim \mathsf{y} \mid \mathsf{x} < \mathsf{y} \mid \mathsf{x} = \mathsf{y}+1 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists \mathsf{x}\, \varphi$

- Last position is labelled by the letter $a \in \Sigma$:

$$\exists \mathsf{x}\ (\neg \exists \mathsf{y}\ \mathsf{x} < \mathsf{y}) \wedge a(\mathsf{x})$$

Data words as first-order structures

- Alphabet $\Sigma = \{a_1, \ldots, a_N\}$ and infinite domain $\mathbb{N}$.
- Data word $\sigma = (a_{i_1}, d_1) \cdots (a_{i_K}, d_K)$ is equivalent to

$$(\{1, \ldots, K\}, <, \sim, +1, P_1, \ldots, P_N)$$

  - For $j, j' \in \{1, \ldots, K\}$, $j \sim j'$ iff $d_j = d_{j'}$.
  - For $l \in \{1, \ldots, N\}$, $P_l \stackrel{\text{def}}{=} \{j \in \{1, \ldots, K\} : a_{i_j} = a_l\}$.
- First-order logic can be naturally interpreted over such structures.

Semantics

$$\sigma \models_v a(\mathsf{x}) \quad \overset{\text{def}}{\Leftrightarrow} \quad \Sigma(\mathsf{x}) = a \qquad \text{(letter at position } \mathsf{x}\text{)}$$

$$\sigma \models_v \mathsf{x} \sim \mathsf{y} \quad \overset{\text{def}}{\Leftrightarrow} \quad \mathbb{N}(\mathsf{x}) = \mathbb{N}(\mathsf{y}) \quad \text{(data at positions)}$$

$$\sigma \models_v \mathsf{x} < \mathsf{y} \quad \overset{\text{def}}{\Leftrightarrow} \quad v(\mathsf{x}) < v(\mathsf{y})$$

$$\sigma \models_v \mathsf{x} = \mathsf{y} + 1 \quad \overset{\text{def}}{\Leftrightarrow} \quad v(\mathsf{x}) = v(\mathsf{y}) + 1$$

$$\sigma \models_v \exists\, \mathsf{x}\, \varphi \quad \overset{\text{def}}{\Leftrightarrow} \quad \text{there is position } i \text{ s.t. } \sigma \models_{v[\mathsf{x} \mapsto i]} \varphi.$$

FO2 and VASS

- Theorem: Satisfiability problem for $\mathrm{FO2}(\sim, <, +1)$ is decidable. [Bojańczyk et al., LICS 06]

- Proof in two steps:
  - Satisfiability is first reduced to nonemptiness for data automata (undefined herein).
  - Nonemptiness for data automata is then reduced to the reachability problem for VASS.

- Theorem: There is a reduction from the reachability problem for VASS to finitary satisfiability for $\mathrm{FO2}(\sim, <, +1)$.

Fixing a few more things (proof)

- Instance: $\mathcal{S} = (Q, n, \delta)$, $(q_i, \vec{0})$, $(q_f, \vec{0})$.
- $\Sigma = Q \uplus \{\mathrm{inc}(i), \mathrm{dec}(i) : i \in [1, n]\}$.
  (below $a \in \{\mathrm{inc}(i), \mathrm{dec}(i) : i \in [1, n]\}$)
- The run $(q_0, \vec{x_0}) \xrightarrow{a_0} (q_1, \vec{x_1}) \xrightarrow{a_1} \cdots \xrightarrow{a_{K-1}} (q_K, \vec{x_K})$ encoded by a data word with projection $q_0 a_0 q_1 a_1 \cdots a_{K-1} q_K$.
- Run

$$\begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ \binom{0}{0} & \binom{1}{0} & \binom{2}{0} & \binom{2}{1} & \binom{1}{1} & \binom{0}{1} & \binom{0}{0} \end{array}$$

  corresponds to data word

$$\begin{array}{ccccccccccccc} q_0 & \mathrm{inc}(1) & q_1 & \mathrm{inc}(1) & q_2 & \mathrm{inc}(2) & q_3 & \mathrm{dec}(1) & q_4 & \mathrm{dec}(1) & q_5 & \mathrm{dec}(2) & q_6 \\ \star & k_1 & \star & k_2 & \star & k_3 & \star & k_1 & \star & k_2 & \star & k_3 & \star \end{array}$$

Enforcing the projection

- φ_{proj}: conjunction of the formulae below.
  - The first letter is q_i:
    $$\exists \mathrm{x}\ (\neg \exists \mathrm{y}\ \mathrm{y} < \mathrm{x}) \wedge q_i(\mathrm{x})$$
  - The last letter is q_f:
    $$\exists \mathrm{x}\ (\neg \exists \mathrm{y}\ \mathrm{x} < \mathrm{y}) \wedge q_f(\mathrm{x})$$
  - Sequence of locations/actions respects graph of $\mathcal{S}$:
    $$\forall\, \mathrm{x}\ (\bigvee_{q \in Q} q(\mathrm{x})) \Rightarrow ((\neg \exists \mathrm{y}\ \mathrm{x} < \mathrm{y}) \vee$$
    $$\bigvee_{q \xrightarrow{a} q' \in \delta} (q(\mathrm{x}) \wedge (\exists \mathrm{y}\ \mathrm{y} = \mathrm{x} + 1 \wedge a(\mathrm{y})) \wedge$$
    $$(\exists \mathrm{y}\ \mathrm{y} = \mathrm{x} + 1 \wedge (\exists \mathrm{x}\ \mathrm{x} = \mathrm{y} + 1 \wedge q'(\mathrm{x})))))$$
- Observe the nice (and standard) recycling of variables.

Constraints on data values

- To encode counter values, each increment or decrement is attached to a datum.

- A desirable data word:

q_0	$\mathrm{inc}(1)$	q_1	$\mathrm{inc}(1)$	q_2	$\mathrm{inc}(2)$	q_3	$\mathrm{dec}(1)$	q_4	$\mathrm{dec}(1)$	q_5	$\mathrm{dec}(2)$	q_6
$\star$	k_1	$\star$	k_2	$\star$	k_3	$\star$	k_1	$\star$	k_2	$\star$	k_3	$\star$

- φ: conjunction of φ_{proj} and formulae below.

  - For $i, j \in [1, n]$, there are no two positions labelled by $\mathrm{inc}(i)$ and $\mathrm{inc}(j)$ having the same datum:

    $$\forall \mathrm{x}\ \mathrm{y}\ (\mathrm{x} < \mathrm{y} \wedge \mathrm{inc}(i)(\mathrm{x}) \wedge \mathrm{inc}(j)(\mathrm{y})) \Rightarrow \neg(\mathrm{x} \sim \mathrm{y}).$$

    (remember $\mathrm{inc}(i)$ and $\mathrm{dec}(i)$ are also letters in Σ)

  - Same with $\mathrm{dec}(i)$ and $\mathrm{dec}(j)$:

    $$\forall \mathrm{x}\ \mathrm{y}\ (\mathrm{x} < \mathrm{y} \wedge \mathrm{dec}(i)(\mathrm{x}) \wedge \mathrm{dec}(j)(\mathrm{y})) \Rightarrow \neg(\mathrm{x} \sim \mathrm{y}).$$

Constraints on data values (II)

- For $i \in [1, n]$, for every position labelled by $\mathrm{dec}(i)$, there is a past position labelled by $\mathrm{inc}(i)$ with the same data value:

$$\forall \mathsf{x}\ \mathrm{dec}(i)(\mathsf{x}) \Rightarrow (\exists\, \mathsf{y}\ (\mathsf{y} < \mathsf{x}) \wedge (\mathsf{x} \sim \mathsf{y}) \wedge \mathrm{inc}(i)(\mathsf{y}))$$

- Since in the final configuration, any counter value is zero, we impose that for $i \in [1, n]$, for every position labelled by $\mathrm{inc}(i)$, there is a future position labelled by $\mathrm{dec}(i)$ with same data value:

$$\forall \mathsf{x}\ \mathrm{inc}(i)(\mathsf{x}) \Rightarrow (\exists\, \mathsf{y}\ (\mathsf{x} < \mathsf{y}) \wedge (\mathsf{x} \sim \mathsf{y}) \wedge \mathrm{dec}(i)(\mathsf{y}))$$

- One can show $(q_f, \vec{0})$ is reachable from $(q_i, \vec{0})$ iff φ is satisfiable.

$\mathrm{FO3}(\sim, <, +1)$ is undecidable [Bojańczyk et al., LICS 06]

- Extend VASS with zero-tests.
- Nonemptiness problem (or equivalent control state reachability) is undecidable.
- Use the third variable to encode zero-tests:

$$\forall \mathsf{x}\ \mathrm{zero}(i)(\mathsf{x}) \Rightarrow$$

$$(\forall \mathsf{y}\ (\mathsf{y} < \mathsf{x} \wedge \mathrm{inc}(i)(\mathsf{y})) \Rightarrow \exists \mathsf{z}\ ((\mathsf{y} < \mathsf{z} < \mathsf{x}) \wedge \mathrm{dec}(i)(\mathsf{y}) \wedge (\mathsf{y} \sim \mathsf{z})))$$

Formalisms for Data Words – Automata

Specifying classes of data words

- Register automata
  - Register automata [Kaminski & Francez, TCS 94]
  - Data automata [Bouyer & Petit & Thérien, IC 03]
  - Machines for strings over infinite alphabets. [Neven & Schwentick & Vianu, TOCL 04]
  - See the survey [Segoufin, CSL'06]

- Many new classes
  - Class automata [Bojańzcyk & Lasota, LICS'10].
  - Variable automata. [Grumberg & Kupferman & Sheinval, LATA'10]
  - etc.
- Many other formalisms
  - Rewriting systems with data [Bouajjani et al., FCT'07]
  - Hybrid logics [Schwentick & Weber, STACS'07]
  - XPath on data trees.

Other relationships with counter automata

- Class counting automata counts the number of data values along a word. [Manuel & Ramanujan, RP'09]
  - Comparisons to constant values.
  - Nonemptiness reduces to the covering problem for VASS.
  - EXPSPACE upper bound with integers in unary.

- Automata for bounded-depth data trees:
  - Decidability of nonemptiness problem by reduction into priority CA. [Björklund & Bojańczyk, ICALP'07]
  - Nonemptiness for priority CA shown decidable in [Reinhardt, Hab. thesis 05].

- Safety fragment of $\mathrm{LTL}^{\downarrow}$ with one register on infinite data words. [Lazić, FST&TCS'06]
  - No U-subformulae with positive polarity.
  - EXPSPACE upper bound by introducing a subclass of gainy counter automata.

What about languages?

- Each formula defines a class of data words (those satisfied by the formula) and a class of words over a finite alphabet obtained by projection.
- Each data logic defines a class of languages made of words over a finite alphabet.
- Each counter automaton (with alphabet and augmented with initial and final control states) defines a class of languages made of words over a finite alphabet.
- $\mathrm{LTL}^{\downarrow}$ with a unique register is equivalent to gainy counter automata. [Demri & Lazić, TOCL 09]
- $\mathrm{FO2}(\sim, <, +1)$ is equivalent to VASS. [Bojańczyk et al., LICS'06]
- See new relationships in [Bojańczyk & Lasota, LICS'10].

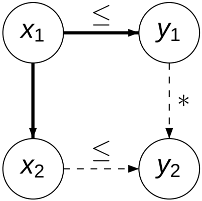
Perspectives

Branching extensions

- Data trees
  - See e.g., register automata for data trees in [Jurdzińsky & Lazić, LICS'07; Figueira, PODS'09]
- Branching VASS: computations are finite trees instead linear structures. [Verma & Goubault-Larrecq, DTMCS 05]
  - Reachability problem for BVASS can be reduced to satisfiability for FO2 over data trees. [Bojańczyk et al., PODS'06]
  - Decidability status of the reachability problem is open.
  - Covering problem and boundedness problem are 2EXPTIME-complete. [Demri et al., FST&TCS'09]

Well-structured transition systems

- Well-structured transition system $(S, \rightarrow, \leq)$
  [Finkel & Schnoebelen, TCS 01]
  - $\leq$ is a well-quasi-ordering: for any infinite sequence $\vec{x_0}, \vec{x_1}, \ldots$ in S, there are $i < j$ such that $\vec{x_i} \leq \vec{x_j}$.
  - $\rightarrow$ and $\leq$ are upward compatible:



- Most decidability proofs uses the well-structuredness of underlying transition systems.

- This is made explicit in [Figueira, ICDT'10].

- How far can we use well-structured transition systems to show decidability? What about computational complexity?

Conclusion

- Data logics/automata can benefit from counter systems.
- Verification of counter systems need data logics as specification languages.
- Relationships with other nonclassical logics: product of modal logics, relevance logics, memoryful temporal logics, hybrid logics, memory logics . . .
- Many open problems in decidability, complexity, expressive power of data logics.

  See e.g., recent [David & Libkin & Tan, LPAR'10]
- Study of new automata models:
  - Variable automata.

    [Grumberg & Kupferman & Sheinvald, LATA'10]
  - Register automata with guess and spread.

    [Figueira, ICDT'10]

References

R. Alur and D. Dill.
A theory of timed automata.
Theoretical Computer Science, 126:183–235, 1994.

R. Alur and Th. Henzinger.
A really temporal logic.
Journal of the Association for Computing Machinery, 41(1):181–204, 1994.

P. Abdulla and B. Jonsson.
Verifying programs with unreliable channels.
Information and Computation, 127(2):91–101, 1996.

H. Björklund and M. Bojańczyk.
Bounded depth data trees.
In *ICALP'07*, volume 4596 of *Lecture Notes in Computer Science*, pages 862–874. Springer, 2007.

M. Bojańczyk, C. David, A. Muscholl, Th. Schwentick, and L. Segoufin.
Two-variable logic on data trees and XML reasoning.
In *PODS'06*, pages 10–19. ACM, 2006.

F. Baader and P. Hanschke.
A scheme for integrating concrete domains into concept languages.
In *IJCAI'91*, pages 452–457, 1991.

A. Bouajjani, P. Habermehl, Y. Jurski, and M. Sighireanu.
Rewriting systems with data.
In *FCT'07*, volume 4639 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.

P. Bouyer.
A logical characterization of data languages.
Information Processing Letters, 84(2):75–85, 2002.

P. Bouyer, A. Petit, and D. Thérien.
An algebraic approach to data languages and timed languages.
Information and Computation, 182(2):137–162, 2003.

P. Blackburn and J. Seligman.
Hybrid languages.
Journal of Logic, Language, and Information, 4:251–272, 1995.

C. David.
Analyse de XML avec données non-bornées.
PhD thesis, LIAFA, Université Paris VII, 2009.

S. Demri and R. Gascon.
The effects of bounding syntactic resources on presburger LTL (extended abstract).
In *TIME'07*, pages 94–104. IEEE, 2007.

Z. Dang, O. Ibarra, and P. San Pietro.
Liveness verification of reversal-bounded multicounter machines with a free counter.
In *FSTTCS'01*, volume 2245 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 2001.

S. Demri, M. Jurdziński, O. Lachish, and R. Lazić.
The covering and boundedness problems for branching vector addition systems.
In *FST&TCS'09, Kanpur*. LZI, 2009.

S. Demri and R. Lazić.
LTL with the freeze quantifier and register automata.
ACM Transactions on Computational Logic, 10(3), 2009.

S. Demri, R. Lazić, and D. Nowak.
On the freeze quantifier in constraint LTL: decidability and complexity.
In *TIME'05*, pages 113–121. IEEE, 2005.

S. Demri, R. Lazić, and A. Sangnier.
Model checking freeze LTL over one-counter automata.
In *FOSSACS'08*, volume 4962 of *Lecture Notes in Computer Science*, pages 490–504. Springer, 2008.

C. David, L. Libkin, and T. Tan.
On the satisfiability of two-variable logic over data words.
In *LPAR'10*, Lecture Notes in Computer Science, pages ??–?? Springer, 2010.
to appear.

D. Figueira.
Satisfiability of downward XPath with data equality tests.
In *PODS'09*, pages 197–206. ACM Press, 2009.

D. Figueira.
Forward-XPath and extended register automata on data-trees.
In *ICDT'10*, pages 231–241. ACM Press, 2010.

M. Fitting.
Modal logic between propositional and first-order.
Journal of Logic and Computation, 12(6):1017–1026, 2002.

A. Finkel and Ph. Schnoebelen.
Well-structured transitions systems everywhere!
Theoretical Computer Science, 256(1–2):63–92, 2001.

D. Figueira and L. Segoufin.
Future-looking logics on data words and trees.
In *MFCS'09*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2009.

D. Gabelaia, A. Kurucz, and M. Zakharyaschev F. Wolter.
Non-primitive recursive decidability of products of modal logics with expanding domains.
Annals of Pure and Applied Logic, 142(1–3):245–268, 2006.

O. Grumberg, O. Kupferman, and S. Sheinvald.
Variable automata over infinite alphabets.
In *LATA'10*, volume 6031 of *Lecture Notes in Computer Science*, pages 561–572. Springer, 2010.

V. Goranko.
Temporal logic with reference pointers.
In D. Gabbay and H. J. Ohlbach, editors, *1st International Conference on Temporal Logic*, pages 133–148. Lecture Notes in Artificial Intelligence, Vol. 827. Springer, Berlin, 1994.

P. Habermehl.
On the complexity of the linear-time mu-calculus for Petri nets.
In *ICATPN'97*, volume 1248 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 1997.

P. Habermehl, R. Iosif, and T. Vojnar.
What else is decidable about integer arrays?
In *FOSSACS'08*, volume 4962 of *Lecture Notes in Computer Science*, pages 474–489. Springer, 2008.

O. Ibarra.
Reversal-bounded multicounter machines and their decision problems.
Journal of the Association for Computing Machinery, 25(1):116–133, 1978.

O. Ibarra.
Restricted one-counter machines with undecidable universe problems.
Mathematical Systems Theory, 13(181):181–186, 1979.

O. Ibarra, J. Su, Z. Dang, T. Bultan, and R. Kemmerer.
Counter machines and verification problems.
Theoretical Computer Science, 289(1):165–189, 2002.

M. Jurdzinski and R. Lazić.
Alternation-free modal mu-calculus for data trees.
In *LICS'07*, pages 131–140. IEEE, 2007.

M. Kaminski and N. Francez.
Finite-memory automata.
Theoretical Computer Science, 134(2):329–363, 1994.

R. Kosaraju.
Decidability of reachability in vector addition systems.
In *STOC'82*, pages 267–281, 1982.

R. Lazić.
Safely freezing LTL.
In *FSTTCS'06*, volume 4337, pages 381–392. LNCS, 2006.

A. Lisitsa and I. Potapov.
Temporal logic with predicate λ-abstraction.
In *TIME'05*, pages 147–155. IEEE, 2005.

C. Lutz.
NEXPTIME-complete description logics with concrete domains.
ACM Transactions on Computational Logic, 5(4):669–705, 2004.

E.W. Mayr.
An algorithm for the general petri net reachability problem.
SIAM Journal of Computing, 13(3):441–460, 1984.

S. Mera.
Modal Memory Logics.
PhD thesis, LORIA & U. of Buenos Aires, 2009.

M. Minsky.
Computation: Finite and Infinite Machines.
Prentice Hall, Englewood Cliffs, NJ, 1967.

A. Manuel and R. Ramanujan.
Counting multiplicity over infinite alphabets.
In *RP'09*, volume 5797 of *Lecture Notes in Computer Science*, pages 141–153. Springer, 2009.

F. Neven, T. Schwentick, and V. Vianu.
Finite state machines for strings over infinite alphabets.
ACM Transactions on Computational Logic, 5(3):403–435, 2004.

J. Ouaknine and J. Worrell.
On the decidability of metric temporal logic.
In *LICS'05*, pages 188–197. IEEE, 2005.

J. Ouaknine and J. Worrell.
On Metric temporal logic and faulty Turing machines.
In *FOSSACS*, volume 3921 of *Lecture Notes in Computer Science*, pages 217–230. Springer, 2006.

K. Reinhardt.
Counting as method, model and task in theoretical computer science.
Habilitation thesis, 2005.

P. Sistla and E. M. Clarke.
The complexity of propositional linear temporal logics.
Journal of the Association for Computing Machinery, 32(3):733–749, 1985.

Ph. Schnoebelen.
Verifying lossy channel systems has nonprimitive recursive complexity.
Information Processing Letters, 83(5):251–261, 2002.

Ph. Schnoebelen.
Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets.
In *MFCS'10*, Lecture Notes in Computer Science. Springer, 2010.
To appear.

L. Segoufin.
Automata and logics for words and trees over an infinite alphabet.
In *CSL'06*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006.

Th. Schwentick and V. Weber.
Bounded-variable fragments of hybrid logics.
In *STACS'07*, volume 4393 of *Lecture Notes in Computer Science*, pages 561–572. Springer, 2007.

A. Urquhart.
The Complexity of Decision Procedures in Relevance Logic II.
The Journal of Symbolic Logic, 64(4):1774–1802, 1999.

K. N. Verma and Jean Goubault-Larrecq.
Karp-Miller Trees for a Branching Extension of VASS.
Discrete Mathematics and Theoretical Computer Science, 7:217–230, 2005.