Outline

Logic in computer science
  Formal verification
  Querying languages
  Modal languages
Overview
  Varia around LTL
  LTL over concrete domains
  Regularity constraints

Memoryful linear-time logics
  LTL over concrete domains
  A binding mechanism
  Main results
Graph constraints
  Path constraints
  Presburger modal logic
A selection of perspectives
  Counter automata
  Semistructured data
  Programs with pointers
Conclusion
Logic in computer science
Verification at the heart of computer science

- Digital systems are everywhere.
  Desktops, embedded systems, cellular phones, etc.
Verification at the heart of computer science

- Digital systems are everywhere. Desktops, embedded systems, cellular phones, etc.

- Needs for verifying functional/security properties:
  - Hardware components
  - Software (programs, communication protocols, web applications, ...)

Verification at the heart of computer science
Verification at the heart of computer science

- Digital systems are everywhere. Desktops, embedded systems, cellular phones, etc.

- Needs for verifying functional/security properties:
  - Hardware components
  - Software (programs, communication protocols, web applications, ...)

*Formal verification is a process in which mathematical techniques are used to guarantee the correctness of a design with respect to some specified behavior.*

[Halpern et al., BSL 01]
FROM SYSTEMS TO MODELS

- Systems are modelled as abstract operational models (counter automata, timed automata, etc.).
**FROM SYSTEMS TO MODELS**

- Systems are modelled as abstract operational models (counter automata, timed automata, etc.).

![Diagram](image-url)

$q_1 \xrightarrow{x = y = 0, \text{lift?}} q_2 \xrightarrow{\text{dial?}} q_3 \xrightarrow{x > 0, \text{connected?}} q_4$

$x = y, x' = y' = 0$

$q_6 \xrightarrow{\text{hang?}} q_5 \xrightarrow{y \leq x}$

$y' \leq x, y++$, quarter!
**Verification as a logical problem**

- Properties are represented by logical formula.
  “The system $S$ never reaches a bad state” becomes $\forall G \neg \text{bad}$. 
Verification as a logical problem

- Properties are represented by logical formula.
  “The system $S$ never reaches a bad state” becomes $\forall G \neg \text{bad}$. 

- Logical problems involve abstract models and formulae.
Verification as a logical problem

- Properties are represented by logical formula. “The system $S$ never reaches a bad state” becomes $\forall G \neg \text{bad}$.

- Logical problems involve abstract models and formulae.

- Development of procedures to solve these problems. Automata, analytic proof systems, ad-hoc methods . . .
**Verification as a logical problem**

- Properties are represented by logical formula. “The system $S$ never reaches a bad state” becomes $\forall G \neg \text{bad}$.

- Logical problems involve abstract models and formulae.

- Development of procedures to solve these problems. Automata, analytic proof systems, ad-hoc methods ... 

- Ultimate goal: automatic verification.
Veriﬁcation as a Logical Problem

- Properties are represented by logical formula.
  “The system $S$ never reaches a bad state” becomes $\forall G \neg \text{bad}$.

- Logical problems involve abstract models and formulae.

- Development of procedures to solve these problems.
  automata, analytic proof systems, ad-hoc methods . . .

- Ultimate goal: automatic veriﬁcation.

- There are theoretical limits for this entreprise.
  - The halting problem for Turing machines is undecidable.

[Turing, 37]
Verification as a logical problem

- Properties are represented by logical formula. “The system $S$ never reaches a bad state” becomes $\forall G \neg \text{bad}$.

- Logical problems involve abstract models and formulae.

- Development of procedures to solve these problems. automata, analytic proof systems, ad-hoc methods …

- Ultimate goal: automatic verification.

- There are theoretical limits for this entreprise.
  - The halting problem for Turing machines is undecidable. [Turing, 37]
  - The set of valid first-order formulae is undecidable. [Church, JSL 36]
**Methodology**

- System, property $\mapsto$ model, logical formula.
**Methodology**

- System, property $\mapsto$ model, logical formula.

- Logical problems:
  - Decision problems (model-checking, validity, satisfiability, . . .)
  - Search problems (controller synthesis, query checking, . . .)
**Methodology**

- System, property $\mapsto$ model, logical formula.

- Logical problems:
  - Decision problems (model-checking, validity, satisfiability, \ldots)
  - Search problems (controller synthesis, query checking, \ldots)

- Analysis of the computational resources to solve the problems
  - Decision procedures vs. undecidability.
  - Complexity in time or memory space.
Methodology

- System, property $\mapsto$ model, logical formula.

- Logical problems:
  - Decision problems (model-checking, validity, satisfiability, . . . )
  - Search problems (controller synthesis, query checking, . . . )

- Analysis of the computational resources to solve the problems
  - Decision procedures vs. undecidability.
  - Complexity in time or memory space.

- Classification
  - Generalizing the models or logics (e.g., ETL)
  - Fragments with better computational properties (e.g., FO2)
Methodology

- System, property $\mapsto$ model, logical formula.

- Logical problems:
  - Decision problems (model-checking, validity, satisfiability, ...)
  - Search problems (controller synthesis, query checking, ...)

- Analysis of the computational resources to solve the problems
  - Decision procedures vs. undecidability.
  - Complexity in time or memory space.

- Classification
  - Generalizing the models or logics (e.g., ETL)
  - Fragments with better computational properties (e.g., FO2)
  - Variants such as fragments of generalizations (e.g., one-clock ATA)
Model-checking and temporal logic

- Temporal logic for specifying behaviors of reactive systems.
  [Pnueli, FOCS 77]
Model-checking and temporal logic

- Temporal logic for specifying behaviors of reactive systems. [Pnueli, FOCS 77]

- Model-checking approach:
  - Computer system is modelled as a graph/model \( M \).
  - Specification is a temporal logic formula \( \phi \).
  - Check whether \( M \) satisfies \( \phi \) \( (M \models \phi) \).
Model-checking and temporal logic

- Temporal logic for specifying behaviors of reactive systems.
  \[ \text{[Pnueli, FOCS 77]} \]

- Model-checking approach:
  - Computer system is modelled as a graph/model $\mathcal{M}$.
  - Specification is a temporal logic formula $\phi$.
  - Check whether $\mathcal{M}$ satisfies $\phi$ ($\mathcal{M} \models \phi$).

- Automata-based approach
  (Gödel prize 2000) \[ \text{[Vardi & Wolper, IC 94]} \]
Model-checking and temporal logic

- Temporal logic for specifying behaviors of reactive systems.
  [Pnueli, FOCS 77]

- Model-checking approach:
  - Computer system is modelled as a graph/model $M$.
  - Specification is a temporal logic formula $\phi$.
  - Check whether $M$ satisfies $\phi$ ($M \models \phi$).

- Automata-based approach
  (Gödel prize 2000) [Vardi & Wolper, IC 94]

- Early work on logic and automata.
  [Büchi, 62]
ANOTHER TRACK ABOUT DATA

Data should be properly formatted:

- Exchanged XML documents on the web.
- Data exchanges between programs/protocols.
Another track about data

Data should be properly formatted:
- Exchanged XML documents on the web.
- Data exchanges between programs/protocols.

```xml
<bibliography name="HDR-D">
```
**Tree representation**

```
HDR-D
  name
  bibliography
     book
        book
            title
              author
               ... 
               ... 
               year
            publisher
            CUP
            2001
            Orwell
            Gal.
            1950
```
Semistructured data:
  - Relaxation of classical relational model.
  - Schema-less (but need for delineating the meaningful data).

Examples:
  - XML documents
  - Web pages with hypertext links

Great variety of models/graphs:
  - Trees vs Graphs
  - Ordered vs Unordered / Ranked vs Unranked trees
  - With vs without data
Logics and reasoning tasks

- Reasoning tasks:
  - Querying (model-checking)
    - integrity constraints (path constraints)
    - type constraints (membership problem for regular tree languages)
  - Comparing type constraints for query optimization
    - implication of path constraints
    - equivalence between tree automata

- Languages
  - XML Path language XPath for addressing part of an XML document.
  - MSO, modal $\mu$-calculus
  - first-order logic
  - modal languages [Alechina, TR 97; Calvanese & de Giacomo & Lenzerini, AAAI 98]

[Bojańczyk et al., PODS 06]
Effectiveness of logic in CS: other areas

[Halpern et al., BSL 01]

Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.

[Davis, OUP 88].
Effectiveness of logic in CS: other areas

[Halpern et al., BSL 01]

*Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.*

[Davis, OUP 88].

- Descriptive complexity: logic and complexity classes.
Effectiveness of logic in CS: other areas

[Halpern et al., BSL 01]

Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.

[Davis, OUP 88].

▶ Descriptive complexity: logic and complexity classes.

▶ Logic as a database query language: SQL is a syntactic variant of first-order logic.
Effectiveness of logic in CS: other areas

[Halpern et al., BSL 01]

*Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.*

[Davis, OUP 88].

- Descriptive complexity: logic and complexity classes.
- Logic as a database query language: SQL is a syntactic variant of first-order logic.
- Reasoning about knowledge, see e.g. [Hintikka, CUP 62].
Effectiveness of logic in CS: other areas

[Halpern et al., BSL 01]

*Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.*

[Davis, OUP 88].

- Descriptive complexity: logic and complexity classes.
- Logic as a database query language: SQL is a syntactic variant of first-order logic.
- Reasoning about knowledge, see e.g. [Hintikka, CUP 62].
- etc.
Modal logic

- Modal languages  [Blackburn & de Rijke & Venema, CUP 01]
  - Simple and sufficiently expressive to talk about relational structures.
  - Local view for the description of structures.

- Applications domains:
  - formal verification: temporal logics, . . .
  - knowledge representation: description and epistemic logics . . .
  - mathematics: arithmetics, . . .
  - linguistics
<table>
<thead>
<tr>
<th>Logic in computer science</th>
<th>Memoryful linear-time logics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>LTL over concrete domains</td>
</tr>
<tr>
<td>Varia around LTL</td>
<td>A binding mechanism</td>
</tr>
<tr>
<td>LTL over concrete domains</td>
<td>Main results</td>
</tr>
<tr>
<td>Regularity constraints</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A selection of perspectives</td>
</tr>
<tr>
<td></td>
<td>Counter automata</td>
</tr>
<tr>
<td></td>
<td>Semistructured data</td>
</tr>
<tr>
<td></td>
<td>Programs with pointers</td>
</tr>
<tr>
<td>Conclusion</td>
<td></td>
</tr>
</tbody>
</table>

**Overview**

- Logic in computer science
- Formal verification
- Querying languages
- Modal languages

**Varia around LTL**

- LTL over concrete domains
- A binding mechanism
- Main results

**Graph constraints**

- Path constraints
- Presburger modal logic

**Regular constraints**

- Counter automata
- Semistructured data
- Programs with pointers

**Conclusion**
Varia around LTL

• Model-checking and satisfiability for LTL fragments.

[D. & Schnoebelen, IC 02]
VARIA AROUND LTL

- Model-checking and satisfiability for LTL fragments.  
  [D. & Schnoebelen, IC 02]

- Model-checking for PC + Wolper-like operator 
  \( \{a^n \cdot b \cdot c^n \cdot d : n \geq 0\} \) is \( \Sigma_1 \)-complete.
Varia around LTL

- Model-checking and satisfiability for LTL fragments.  
  [D. & Schnoebelen, IC 02]

- Model-checking for PC + Wolper-like operator  
  \( \{a^n \cdot b \cdot c^n \cdot d : n \geq 0\} \) is \( \Sigma_1 \)-complete.

- Parameterized complexity for symbolic model-checking.  
  [D. & Laroussinie & Schnoebelen, JSCC 06]
Varia around LTL

- Model-checking and satisfiability for LTL fragments.
  \[ \text{[D. & Schnoebelen, IC 02]} \]

- Model-checking for PC + Wolper-like operator
  \[ \{a^n \cdot b \cdot c^n \cdot d : n \geq 0\} \text{ is } \Sigma_1^1\text{-complete.} \]

- Parameterized complexity for symbolic model-checking.
  \[ \text{[D. & Laroussinie & Schnoebelen, JSCC 06]} \]

- Automata-based approach for LTL over \(\omega^k\)-sequences.
  \[ \text{[D. & Nowak, IJFCS 06; D. & Rabinovich, Submitted]} \]


**VARIA AROUND LTL**

- Model-checking and satisfiability for LTL fragments.
  
  [D. & Schnoebelen, IC 02]

- Model-checking for PC + Wolper-like operator

  \[ \{ a^n \cdot b \cdot c^n \cdot d : n \geq 0 \} \text{ is } \Sigma_1^1 \text{-complete.} \]

- Parameterized complexity for symbolic model-checking.
  
  [D. & Laroussinie & Schnoebelen, JSCC 06]

- Automata-based approach for LTL over \( \omega^k \)-sequences.
  
  [D. & Nowak, IJFCS 06; D. & Rabinovich, Submitted]

- LTL properties over admissible counter systems
  
  [Finkel et al., ATVA 06]
**Verification of Qualitative and Quantitative Properties**

- **LTL over concrete domains**  
  [D. & D’Souza, IC 07]

- **Fragments of Presburger LTL.**
  - Decidable fragments.  
  [D. & Gascon, CONCUR 05]
  - Undecidable fragments.  
  [D. & Gascon, TIME 07]

- **Memoryful linear-time temporal logics.**
  - Freeze operator and equality constraints.  
  [D. & Lazić, LICS 06; D. & Lazić & Nowak, IC 07]
  - Decidable version with repeating values.  
  [D. & D’Souza & Gascon, LFCS 07]
  - Model-checking problems.  
  [D. & Lazić & Sangnier, On-going]
Regularity constraints

- Hybrid modal logic for path constraints.  
  [Alechina & D. & de Rijke, JLC 03]

- Presburger modal logic for trees  
  [D. & Lugiez, IJCAR 06]

- Dynamic logic of permission  
  [D., JLC 05]

- Complexity of regular modal logics
  - Translation into guarded fragment with two variables.  
    [D. & de Nivelle, JoLLI 05]
  - PSpace or ExpTime bounds.  
    (many papers)
Selection 1: memoryful linear-time logics
**STANDARD LTL**

- Formulae: $\phi ::= p \mid X\phi \mid \phi U \phi \mid \neg \phi \mid \phi \land \phi$.

- Models: $\sigma : \mathbb{N} \rightarrow \mathcal{P}($PROP$)$.

- Satisfaction relation: $\sigma, i \models \phi$.

$X\phi$: next-time $\phi$

$\phi_1 U \phi_2$: $\phi_1$ until $\phi_2$
**Complexity issues for LTL**

- $\phi \mapsto A_\phi$
  - models of $\phi = L(A_\phi)$.
  - $|A_\phi|$ is in $2^{O(|\phi|)}$.

- Model-checking and satisfiability are $PSPACE$-complete.
  - [Sistla & Clarke, JACM 85]

- Extended temporal logic is also $PSPACE$-complete.
  - [Wolper, IC 83; Vardi & Wolper, IC 94]
LTL OVER CONCRETE DOMAINS

- Constraint system \( \mathcal{D} = (D, (R_\alpha)_{\alpha \in I}) \).
- Examples: \((\mathbb{N}, =, <), (\mathbb{N}, =, \text{succ}), (\mathbb{R}, =, <), (\{0, 1\}^*, <, =)\)
- Atomic constraint: \(R(x_1, \ldots, x_t), x_i \in \text{VAR} \).
LTL OVER CONCRETE DOMAINS

- Constraint system $\mathcal{D} = (D, (R_\alpha)_{\alpha \in I})$.
- Examples: $(\mathbb{N}, =, <), (\mathbb{N}, =, \text{succ}), (\mathbb{R}, =, <), (\{0, 1\}^*, <, =)$
- Atomic constraint: $R(x_1, \ldots, x_t), x_i \in \text{VAR}$.
- Logic CLTL($\mathcal{D}$):
  - Formulae: $\phi ::= R(X^{n_1}x_1, \ldots, X^{n_t}x_t) \mid X\phi \mid \phi U \phi \mid \ldots$
  - Models: $\sigma : \mathbb{N} \to (\text{VAR} \to D)$.
  - $\sigma, j \models R(X^{n_1}x_1, \ldots, X^{n_t}x_t)$ iff
    value of $x_1$ in the $j+n_1$th state
    $(\sigma(j + n_1)(x_1), \ldots, \sigma(j + n_t)(x_t)) \in R$

i.e. values at different states can be compared.
Problems

- **Satisfiability**: given a formula $\phi$, is there a model $\sigma$ such that $\sigma, 0 \models \phi$?

- **Existential model-checking**:

$$Xx = 2$$

$$Xx = x - 1 \quad Xx = x + 1 \quad \models (x = 0) \land GF(x = 0)$$


**ANALOGOUS FORMALISMS**

- Temporal logics with Presburger constraints
  - constraints on the number of event occurrences
    [Bouajjani & Echahed & Habermehl, LICS 95]
  - LTL with counters [Comon & Cortier, CSL 00]

- Description logics with concrete domains
  [Baader & Hanschke, IJCAI 91; Lutz, ToCL 04]

- Spatio-temporal logics
  [Balbiani & Condotta, FROCOS 02; Gabbay et al., book 03]
FRAGMENTS OF PRESBURGER LTL

- Model-checking for \( \text{CLTL}(\mathbb{N}, = 0?, +1) \) is undecidable [Minsky, Book 67].

- Model-checking for \( \text{CLTL}(\mathbb{Z}, <, =, \equiv_k, = d?) \) is \( \text{PSPACE} \)-complete
  
  [D. & D’Souza, IC 07; D. & Gascon, CONCUR 05]

- Branching-time extensions in [Gascon, thesis 07].

- Open problem: decidability status for \( \text{CLTL}(\{0, 1\}^*, <, =) \).
Temporal logics with memory

- Real-time logic TPTL
  
- MTL
  - Fin. MTL is decidable
  - Inf. MTL is undecidable

  [Alur & Henzinger, JACM 94]

  [Koymans, RTS 90]

  [Ouaknine & Worrell, LICS 05]

  [Ouaknine & Worrell, FOSSACS 06]
Temporal logics with memory

- Real-time logic TPTL
  - [Alur & Henzinger, JACM 94]

- MTL
  - Fin. MTL is decidable
  - Inf. MTL is undecidable
  - [Ouaknine & Worrell, LICS 05]
  - [Ouaknine & Worrell, FOSSACS 06]

- LTL with forgettable past
  - [Laroussinie & Markey & Schnoebelen, LICS 02]

\[ p \rightarrow \text{Now } XG^{-1} p, p \rightarrow p \rightarrow \]

\[ p \rightarrow G^{-1} p, p \rightarrow \]
**Freeze quantifier in hybrid logics**

- \( \downarrow_x \phi \): \( \phi \) holds true in the variant model where \( x \) is true only at the current state

  [Blackburn & Seligman, JoLLI 95; Goranko, JoLLI 96].

- Every reachable state can be visited infinitely often:

  \( \forall G \downarrow_x \exists X F x \).
LTL WITH MEMORY

- Formulae:

\[ \phi ::= X^i x = X^j x' \mid \downarrow_{z=X^j y} \phi \mid X\phi \mid \phi U \phi \mid \neg \phi \mid \ldots \]

where \( x, x' \in \text{VAR}_f \cup \text{VAR}_r, \ z \in \text{VAR}_r, \ y \in \text{VAR}_f. \]
LTL with Memory

- Formulae:

\[
\phi ::= X^i x = X^j x' \mid \downarrow z = X^j y \phi \mid X\phi \mid \phi U \phi \mid \neg \phi \mid \ldots
\]

where \( x, x' \in \text{VAR}_f \cup \text{VAR}_r \), \( z \in \text{VAR}_r \), \( y \in \text{VAR}_f \).

- Infinite models: \( \sigma : \mathbb{N} \rightarrow (\text{VAR}_f \rightarrow \mathbb{N}) \).
LTL with memory

- Formulae:

\[ \phi ::= X^i x = X^j x' \mid \downarrow_z X^j_y \phi \mid X\phi \mid \phi U \phi \mid \neg \phi \mid \ldots \]

where \( x, x' \in \text{VAR}_f \cup \text{VAR}_r, z \in \text{VAR}_r, y \in \text{VAR}_f \).

- Infinite models: \( \sigma : \mathbb{N} \rightarrow (\text{VAR}_f \rightarrow \mathbb{N}) \).

- \( \sigma, i \models_{e} \downarrow_z X^j_y \phi \overset{\text{def}}{\iff} \sigma, i \models_{e'} \phi \) with \( e' = e[z \leftarrow \sigma(i + j)(y)] \).
LTL with memory

- Formulae:

\[ \phi ::= X^i x = X^j x' \mid \downarrow_{z=x} X^j y \phi \mid X\phi \mid \phi U \phi \mid \neg \phi \mid \ldots \]

where \( x, x' \in \text{VAR}_f \cup \text{VAR}_r, z \in \text{VAR}_r, y \in \text{VAR}_f \).

- Infinite models: \( \sigma : \mathbb{N} \rightarrow (\text{VAR}_f \rightarrow \mathbb{N}) \).

- Properties
  - Repeating value: \( \downarrow_{z=x} XF(z = x) \).
  - Nonce property: \( G(\downarrow_{z=x} XG(z \neq x)) \).
Undecidability results

- Undecidability:
  - Satisfiability for $\text{LTL}^1(X, U)$ restricted to one flexible variable and two rigid variables is $\Sigma^1_1$-complete.
    
    [D. & Lazić & Nowak, IC 07]
  - By reduction from recurrent reachability problem for nondeterministic Minsky machines.
  - See also [Lisitsa & Potapov, TIME 05]
UNDECIDABILITY RESULTS

- Undecidability:
  - Satisfiability for $\text{LTL}^\downarrow_{1}(X, U)$ restricted to one flexible variable and two rigid variables is $\Sigma^1_1$-complete.  
    ![Undecidability: Satisfiability for $\text{LTL}^\downarrow_{1}(X, U)$ restricted to one flexible variable and two rigid variables is $\Sigma^1_1$-complete.](D. & Lazić & Nowak, IC 07)
  - By reduction from recurrent reachability problem for nondeterministic Minsky machines.
  - See also ![Undecidability: Satisfiability for $\text{LTL}^\downarrow_{1}(X, U)$ restricted to one flexible variable and two rigid variables is $\Sigma^1_1$-complete.](Lisitsa & Potapov, TIME 05)

- Restriction to one rigid variable: ![Restriction to one rigid variable: Satisfiability for $\text{LTL}^\downarrow_{1}(X, U)$ restricted to one flexible variable and to one rigid variable is $\Pi^0_1$-complete.](D. & Lazić, LICS 06)
  - Satisfiability for $\text{LTL}^\downarrow_{1}(X, U)$ restricted to one flexible variable and to one rigid variable is $\Pi^0_1$-complete.
  - By reduction from infinitary nonemptiness for incrementing counter automata (subclass of ICMETs).
A decidability result

- Satisfiability for $\text{LTL}^\downarrow(X, U)$ over finite models restricted to one flexible variable and to one rigid variable is decidable but not primitive recursive.
A decidability result

- Satisfiability for $\text{LTL}^\downarrow(X, U)$ over finite models restricted to one flexible variable and to one rigid variable is decidable but not primitive recursive.

- Decidability proof in two steps:
  1. From formulae to alternating register automata.
  2. From alternating register automata with a unique register to incrementing counter automata.

See also [Lasota & Walukiewicz, FOSSACS 05; Ouaknine & Worrell, LICS 05].
A decidability result

- Satisfiability for $\text{LTL}^\downarrow(X, U)$ over finite models restricted to one flexible variable and to one rigid variable is decidable but not primitive recursive.

- Decidability proof in two steps:
  1. From formulae to alternating register automata.
  2. From alternating register automata with a unique register to incrementing counter automata.

See also [Lasota & Walukiewicz, FOSSACS 05; Ouaknine & Worrell, LICS 05].

- Non primitive recursiveness is also proved in two steps
  1. Finitary nonemptiness for incrementing counter automata is non PR by adapting [Schnoebelen, IPL 02].
  2. This problem can be reduced in logspace to satisfiability in $\text{LTL}^\downarrow(X, U)$ restricted to one rigid variable.
**Summary (with one flexible variable)**

<table>
<thead>
<tr>
<th>Models</th>
<th>Finite</th>
<th>Infinite</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of rigid variables</strong></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>X, F</td>
<td>D, not PR</td>
<td>Σ₁⁻⁰⁻C</td>
</tr>
<tr>
<td>X, U</td>
<td>D, not PR</td>
<td>Σ₁⁻⁰⁻C</td>
</tr>
<tr>
<td>X, F, F⁻¹</td>
<td>Σ₁⁻⁰⁻C</td>
<td>Σ₁⁻⁰⁻C</td>
</tr>
</tbody>
</table>
First-order languages and automata

- Register automata
  - Finite-memory automata [Kaminsky & Francez, TCS 94]
  - Data automata [Bouyer & Petit & Thérien, IC 03]
- See also [(D. & Lazić; Bojańczyk et al.), LICS 06]
First-order languages and automata

- Register automata
  - Finite-memory automata
  - Data automata
  - See also

- First-order languages
  - $\text{FO2}(\sim, <, +1)$ over data words equivalent to reachability in Petri nets.
  - Decidable fragments over data trees.
  - Survey in [Segoufin, CSL 06].
Decidable logic CLTL$\Diamond$ with repeating values

- Formulae:

$$\phi ::= x = X^i y \mid x = \Diamond y \mid \phi \land \phi \mid \neg \phi \mid X \phi \mid \phi U \phi \mid X^{-1} \phi \mid S \phi$$

Stéphane Demri

Logics for Specification and Verification
**Decidable logic CLTL° with repeating values**

- **Formulae:**
  \[
  \phi ::= x = X^i y \mid x = \Diamond y \mid \phi \land \phi \mid \neg \phi \mid X \phi \mid \phi U \phi \mid X^{-1} \phi \mid \phi S \phi
  \]

- **Finitary and infinitary satisfiability for CLTL° is decidable.**
  
  \[\text{[D. & D'Souza & Gascon, LFCS 07]}\]
  
  - By reduction to checking fairness conditions in Petri nets.  
    \[\text{[Jančar, TCS 90]}\]

  - \text{PSPACE}\text{-completeness with a unique flexible variable.}
  
  - Decidability is preserved with MSO-definable operators.
OTHER DECIDABLE PROBLEMS

► Safety fragment:
  ▶ No U in the scope of an even number of negations.
  ▶ Infinitary satisfiability for the safety fragment of CLTL$^\downarrow$($X$, $U$) with one rigid variable, one flexible variable and an alphabet is $\text{ExpSpace}$-complete. [Lazić, FSTTCS 06]
  ▶ Finitary satisfiability has the same complexity as CLTL$^\downarrow$($X$, $U$) with one rigid variable.

► Branching extension:
  ▶ Modal $\mu$-calculus with freeze quantification and one register over finite data trees is decidable. [Jurdziński & Lazić, LICS 07]
  ▶ See also [Bojańczyk et al., PODS 06]
Model-checking problems

[D. & Lazić & Sangnier, On-going]

- Runs of Minsky machines can be viewed as data words:
  \[(q_0, c_0), \ldots, (q_n, c_n)\]

- Model-checking problem over deterministic one-counter
  Minsky machines with $\text{CLTL}^\uparrow(X, U)$ is $\text{PSPACE}$-complete
  (with finite or infinite runs)

- Model-checking problem over non-deterministic one-counter
  Minsky machines with $\text{CLTL}^\uparrow(X, U)$ is undecidable (with
  finite or infinite runs).

- What happens with other operational models (stack
  automata, etc.)?
Selection 2: graph constraints
REASONING TASKS FOR SEMISTRUCTURED DATA

- Querying (model-checking)
  - Integrity constraints, e.g. path constraints \((a \cdot b)^* \subseteq (c \cup e)\).
  
- Type constraints.
  E.g., membership problem for regular tree languages.

- Comparing constraints (validity)
  - Emptiness problem for a Boolean expression built over constraints.
    E.g., implication of path constraints \(a \subseteq b \models a \cdot c \subseteq b \cdot c\),
    equivalence between tree automata.
  
- Comparing integrity constraints given type constraints.
Modal approach

- Schemes subsumption encoded into a hybrid modal logic.
  [Alechina, TR 97]

- Schemes subsumption encoded into a description logic.
  [Calvanese & de Giacomo & Lenzerini, AAAI 98]

- DTD with well-typed references encoded into a hybrid modal logic with binder ↓
  [Bidoit & Cerrito & Thion, JANCL 04]

- Path constraints encoded into fragments of hybrid modal logics.
  [Franceschet & de Rijke, JAL 06]

- XPath queries and equivalence problem encoded into PDL over finite node labelled ordered trees.
  [Marx, TABLEAUX 03]
Path constraints

- Integrity constraints from [Abiteboul & Vianu, PODS 97]

- Interests of regular path expressions:
  - They give semantical information on the data.
  - They are used for query optimization.

- Regular path expressions:
  \[ p ::= a \mid \epsilon \mid p + p \mid p^* \mid p; p \mid \# \].

- Simple path expressions: \[ p ::= a \mid \epsilon \mid p; p \].
Models

- Rooted edge labeled connected graphs:
  - (XML) Documents with pointers (id/idref attributes).
  - Web pages with hyperlinks.

- $\Sigma$-structure: $G = (S, rt, (R_a)_{a \in \Sigma})$

- Deterministic vs non-deterministic structures.
Path constraints

(a) forward constraint  (b) backward const.  (c) lollipop const.
Path constraints

(a) forward constraint  (b) backward const.  (c) lollipop const.

Query evaluation problem for a class $C$ of path constraints:

instance: a finite $\Sigma$-structure $G$ and a constraint $c$ in $C$;

question: $G \models c$?
Path constraints

(a) forward constraint  (b) backward const.  (c) lollipop const.

- Query evaluation problem for a class $C$ of path constraints:
  instance: a finite $\Sigma$-structure $G$ and a constraint $c$ in $C$;
  question: $G \models c$?

- Containment problem for a class $C$ of path constraints:
  instance: constraints $c_1, \ldots, c_{n+1}$, $n \geq 0$, in $C$;
  question: is it the case that for every $\Sigma$-structure $G$, $G \models c_1$ and $\ldots$ and $G \models c_n$ imply $G \models c_{n+1}$?
Some results

- The containment problem for forward constraints is in \textsc{ExpSpace}. \[\text{[Abiteboul & Vianu, PODS 97]}\]

- The containment problem for forward constraints with simple path expressions is in \textsc{PTIME}. \[\text{[Abiteboul & Vianu, PODS 97]}\]

- The containment problem for lollipop constraints with simple path expressions is undecidable. \[\text{[Buneman & Fan & Weinstein, PODS 98]}\]
PDL\textsuperscript{path}

A PDL-like logic to encode problems on standard path constraints.

- Formulae (we allow $p^{-1}$):

$$
\top \mid \bot \mid \text{root} \mid \neg \phi \mid \phi \land \phi \mid [p]\phi \mid \langle p \rangle \phi
$$

- no propositional variables, a unique nominal root.

- Models: Σ-structures

- Satisfiability/validity problem (at the root).
About $\mathsf{PDL}^\text{path}$

- The model checking problem for $\mathsf{PDL}^\text{path}$ is $\mathsf{PTime}$-complete.

- The satisfiability and validity problems for $\mathsf{PDL}^\text{path}$ are in $\mathsf{ExpTime}$ (by translation into CPDL with nominals).

- The satisfiability problem for $\mathsf{PDL}^\text{path}$ is $\mathsf{ExpTime}$-hard whenever $|\Sigma| \geq 1$. 

About $\text{PDL}^\text{path}$

- The model checking problem for $\text{PDL}^\text{path}$ is $\text{PTime}$-complete.

- The satisfiability and validity problems for $\text{PDL}^\text{path}$ are in $\text{ExpTime}$ (by translation into $\text{CPDL}$ with nominals).

- The satisfiability problem for $\text{PDL}^\text{path}$ is $\text{ExpTime}$-hard whenever $|\Sigma| \geq 1$.

- Some consequences:
  - The query evaluation problem for the class of path constraints is $\text{NLogSpace}$-complete both for deterministic and non-deterministic graphs.
  - The containment problem for forward constraints is in $\text{ExpTime}$, while it is at least $\text{PSPACE}$-hard if $|\Sigma| \geq 2$. 

Stéphane Demri

Logics for Specification and Verification
An open problem

Complexity of containment problem for forward constraints and for backward constraints?

- Over multi-root structures the containment problem restricted to instances of the form $c_1, \ldots, c_{n+1}$ such that for $i \in \{1, \ldots, n\}$, $c_i$ is of the form $p_i \subseteq_f q_i$ and $q_i$ is a word, is $\text{PSPACE}$-complete. [Debarbieux, thesis 05]

  - Use of prefix rewriting techniques. [Dauchet & Tison, LICS 90]
Presburger constraints on graphs/trees

- Constraints in counter automata.

- Constraints on the number of event occurrences.
  [Bouajjani & Echahed & Habermehl, LICS 95]

- Constraints on XML documents.
  [Dal Zilio & Lugiez, RTA 03; Seidl et al., ICALP 04]

- Graded modal logics ($\diamondsuit \geq 3 \ p$). [Fine, NDJFL 72]

- Description logics ($\geq 3 \ R \cdot C$). [Hollunder & Baader, KR 91]

- Hennessy-Milner Logic (HML).
Presburger constraints in graphs

\[
\begin{align*}
  u_1 & \models \phi_1 \\
  u_2 & \models \phi_2 \\
  u_3 & \models \phi_1 \land \phi_2 \\
  u_4 & \models \phi_1 \\
\end{align*}
\]

\[
u \models \#\phi_1 = \#\phi_2 + 1.
\]
Logics that count in PSpace

- Minimal graded modal logic. [Tobies, CADE 99]
- Majority logic. [Pacuit & Salame, KR 04]
- Rank-1 modal logics. [Schröder & Pattinson, LICS 06]
- Constraints on sets with cardinalities. [Kuncak & Manette & Rinard, Dagstuhl 05]
Presburger modal logic

- Modal logic with quantifier-free Presburger constraints is \( \text{PSPACE}-\text{complete} \).

[D. & Lugiez, IJCAR 06]
Presburger modal logic

- Modal logic with quantifier-free Presburger constraints is \( \text{PSpace} \)-complete. \([\text{D. & Lugiez, IJCAR 06}]\)

- This \( \text{PSpace} \) upper bound can be preserved with a bit of regularity constraints.
PMODAL LOGIC

- Modal logic with quantifier-free Presburger constraints is PSPACE-complete. [D. & Lugiez, IJCAR 06]

- This PSPACE upper bound can be preserved with a bit of regularity constraints.

- An undecidable extension:
  - Mix of PML and PDLtree. [Afanasiev et al., JANCL 05]
  - Models: finite labeled unranked ordered trees.
  - Relation symbols: Σ = {↓, ↓*, →, →*, ←, ←*, ↑, ↑*}
  - Formulas:
    - \( \phi ::= p \mid \neg \phi \mid \phi \land \phi \mid t \sim b \)
    - \( t ::= a \times \#^R \phi \mid t + a \times \#^R \phi \),
A selection of perspectives: resource logics and verification
A selection of perspectives: resource logics and verification

- Counter automata
- Programs with pointers
- Data logics

Stéphane Demri Logics for Specification and Verification
Verification of counter automata

Broad goal: to determine classes of counter automata with effective verification

- Counter automata and data logics.

- Admissible counter systems and MSO properties.

- Complexity of reachability problems for subclasses of counter automata.
Querying semistructured data

Broad goal: to design algorithms to query documents with data over enriched languages

- What are the tractable fragments of branching-time memoryful temporal logics.
- Presburger constraints on XML documents with data.
- Complexity issues for problems with path constraints.
Reasoning about programs with pointers

Broad goal: to design temporal languages to specify the behaviors of pointer programs

- To combine an assertion language from separation logic with linear-time/branching-time temporal logics. See e.g., [Brochenin & D. & Lozes, LFCS 07]
- Analysis of high-level properties on abstract models.
- Comparison with existing logical formalisms for memory?
  - Pointer assertion logic (WS2S + invariants). [Jensen et al. 97]
  - TVLA (3-valued logic). [Lev-Ami & Sagiv, SAS’00]
  - Logic of Reachable Patterns. [Yorsh et al., FOSSACS’06]
  - Evolution Logic [Yahav et al., ESOP’03]
CONCLUDING REMARKS

- Ubiquity of logical formalisms
  - Formal verification of computer systems.
  - Database query languages.
  - Reasoning about knowledge.

- A grand challenge

  Security issues on Internet:
  Combining Model-checking & Databases Theory.