Automata-logic connection for guarded logics

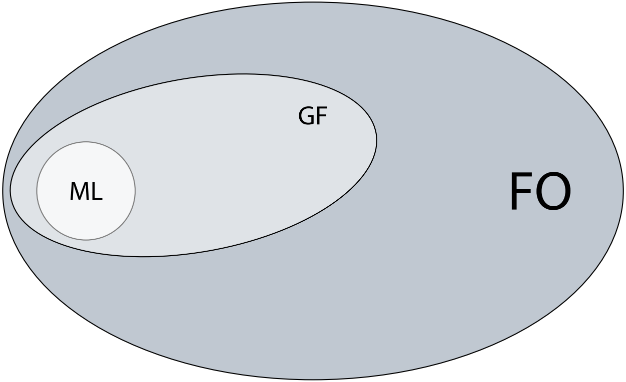
Michael Vanden Boom

University of Oxford

CiE 2015 - Special Session on Automata, Logic, and Infinite Games
July 2015

Including joint work with
Michael Benedikt, Balder ten Cate, and Thomas Colcombet

Guarded logics

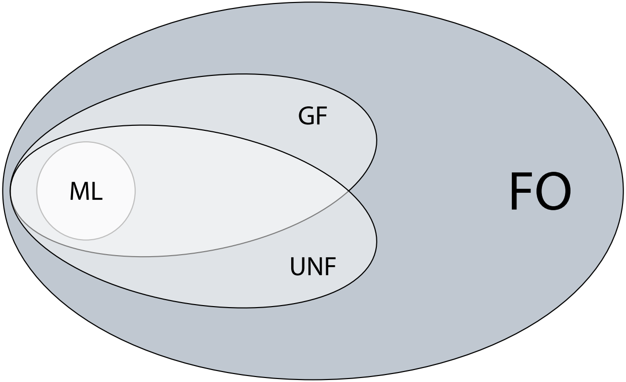


constrain quantification

$$\exists \boldsymbol{x}(G(\boldsymbol{xy}) \wedge \psi(\boldsymbol{xy}))$$
$$\forall \boldsymbol{x}(G(\boldsymbol{xy}) \rightarrow \psi(\boldsymbol{xy}))$$

[Andréka, van Benthem, Németi ’95-’98]

Guarded logics



constrain quantification

$\exists \boldsymbol{x}(G(\boldsymbol{xy}) \wedge \psi(\boldsymbol{xy}))$
$\forall \boldsymbol{x}(G(\boldsymbol{xy}) \rightarrow \psi(\boldsymbol{xy}))$

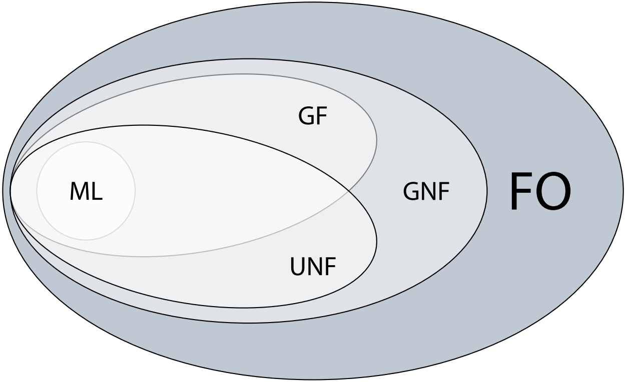
[Andréka, van Benthem, Németi '95-'98]

constrain negation

$\exists \boldsymbol{x}(\psi(\boldsymbol{xy}))$
$\neg\psi(x)$

[ten Cate, Segoufin '11]

Guarded logics



constrain quantification

$\exists \boldsymbol{x}(G(\boldsymbol{xy}) \wedge \psi(\boldsymbol{xy}))$
$\forall \boldsymbol{x}(G(\boldsymbol{xy}) \rightarrow \psi(\boldsymbol{xy}))$

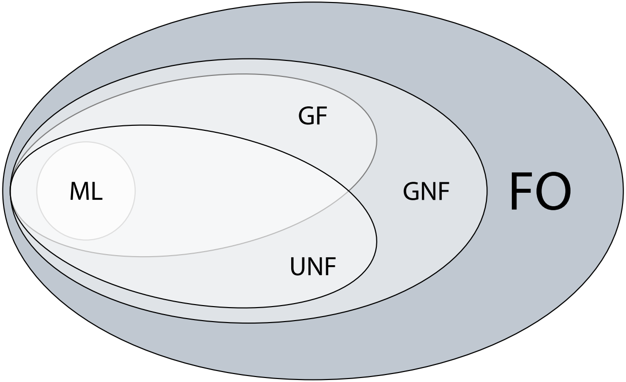
[Andréka, van Benthem, Németi '95-'98]

constrain negation

$\exists \boldsymbol{x}(\psi(\boldsymbol{xy}))$
$G(\boldsymbol{xy}) \wedge \neg\psi(\boldsymbol{xy})$

[ten Cate, Segoufin '11]
[Bárány, ten Cate, Segoufin '11]

Guarded logics



constrain quantification

$\exists \boldsymbol{x}(G(\boldsymbol{xy}) \wedge \psi(\boldsymbol{xy}))$
$\forall \boldsymbol{x}(G(\boldsymbol{xy}) \rightarrow \psi(\boldsymbol{xy}))$

[Andréka, van Benthem, Németi ’95-’98]

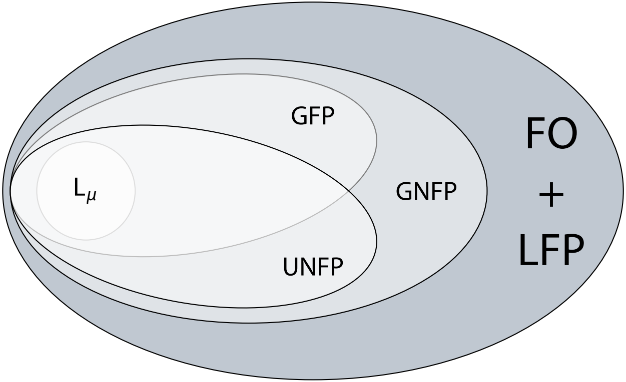
constrain negation

$\exists \boldsymbol{x}(\psi(\boldsymbol{xy}))$
$G(\boldsymbol{xy}) \wedge \neg\psi(\boldsymbol{xy})$

[ten Cate, Segoufin ’11]
[Bárány, ten Cate, Segoufin ’11]

Guarded logics extend modal logic
while still retaining many of its nice properties, e.g. **decidable satisfiability**.

Guarded logics



constrain quantification

$\exists \boldsymbol{x}(G(\boldsymbol{xy}) \wedge \psi(\boldsymbol{xy}))$
$\forall \boldsymbol{x}(G(\boldsymbol{xy}) \rightarrow \psi(\boldsymbol{xy}))$

[Andréka, van Benthem, Németi '95-'98]

constrain negation

$\exists \boldsymbol{x}(\psi(\boldsymbol{xy}))$
$G(\boldsymbol{xy}) \wedge \neg\psi(\boldsymbol{xy})$

[ten Cate, Segoufin '11]
[Bárány, ten Cate, Segoufin '11]

These guarded fixpoint logics extend the modal μ-calculus while still retaining many of its nice properties, e.g. **decidable satisfiability**.

Exploiting model theoretic properties of these guarded logics

GF, UNF, and GNF have **finite model property**
(but fixpoint extensions do not).

Exploiting model theoretic properties of these guarded logics

GF, UNF, and GNF have **finite model property**
(but fixpoint extensions do not).

GFP, UNFP, and GNFP have **tree-like models**
(models of bounded tree-width).

Exploiting model theoretic properties of these guarded logics

GF, UNF, and GNF have **finite model property**
(but fixpoint extensions do not).

GFP, UNFP, and GNFP have **tree-like models**
(models of bounded tree-width).

⇒ amenable to techniques using tree automata

The plan for this talk

Construct automata for deciding **satisfiability** of GFP sentences. [Grädel+Walukiewicz '99]

Describe how these automata can be adapted to decide certain **boundedness** problems. [Benedikt, Colcombet, ten Cate, VB. '15]

Guarded fixpoint logic (GFP)

Fix some relational signature σ.

Syntax for GFP$[\sigma]$

$$\begin{aligned}\varphi \;::=\; & R\mathbf{x} \mid \neg R\mathbf{x} \mid Y\mathbf{x} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists \mathbf{y}(G(\mathbf{xy}) \wedge \varphi(\mathbf{xy})) \mid \forall \mathbf{y}(G(\mathbf{xy}) \rightarrow \varphi(\mathbf{xy})) \mid \\ & [\mathbf{lfp}_{Y,\mathbf{y}}.\varphi(\mathbf{y}, Y, \mathbf{Z})](\mathbf{x}) \text{ where } Y \text{ only occurs positively in } \varphi \mid \\ & [\mathbf{gfp}_{Y,\mathbf{y}}.\varphi(\mathbf{y}, Y, \mathbf{Z})](\mathbf{x}) \text{ where } Y \text{ only occurs positively in } \varphi\end{aligned}$$

where R is a relation in σ or $=$, and
the **guards** $G(\mathbf{xy})$ are atomic formulas that use all of the variables $\mathbf{xy}$.

Guarded fixpoint logic (GFP)

Fix some relational signature σ.

Syntax for GFP$[\sigma]$

$$\varphi \ ::= \ R\boldsymbol{x} \ | \ \neg R\boldsymbol{x} \ | \ Y\boldsymbol{x} \ | \ \varphi \wedge \varphi \ | \ \varphi \vee \varphi \ | \ \exists \boldsymbol{y}(G(\boldsymbol{xy}) \wedge \varphi(\boldsymbol{xy})) \ | \ \forall \boldsymbol{y}(G(\boldsymbol{xy}) \rightarrow \varphi(\boldsymbol{xy})) \ |$$
$$[\mathbf{lfp}_{Y,\boldsymbol{y}}.\varphi(\boldsymbol{y}, Y, \boldsymbol{Z})](\boldsymbol{x}) \text{ where } Y \text{ only occurs positively in } \varphi \ |$$
$$[\mathbf{gfp}_{Y,\boldsymbol{y}}.\varphi(\boldsymbol{y}, Y, \boldsymbol{Z})](\boldsymbol{x}) \text{ where } Y \text{ only occurs positively in } \varphi$$

where R is a relation in σ or $=$, and
the **guards** $G(\boldsymbol{xy})$ are atomic formulas that use all of the variables $\boldsymbol{xy}$.

Examples

$$\varphi_1(x) := \forall y(Sxy \rightarrow \exists z(Ryz \wedge Py \wedge Pz))$$
$$\varphi_2 := \forall x(\exists y(Rxy \wedge \neg Ryx)) \ \equiv \ \forall x(x = x \rightarrow (\exists y(Rxy \wedge \neg Ryx))$$
$$\varphi_3(y) := [\mathbf{lfp}_{Y,y} \,.\, Py \vee \exists z(Ryz \wedge Yz)](y)$$

Tree-like models for GFP

Theorem (Grädel '99)

Every satisfiable $\varphi \in \mathrm{GFP}$ of width k has a model of tree width at most $k - 1$.

Tree-like models for GFP

Theorem (Grädel '99)

Every satisfiable $\varphi \in$ GFP of width k has a model of tree width at most $k-1$.

A structure $\mathfrak{A}$ has **tree width** $k-1$ if it can be covered by (overlapping) bags of size at most k, arranged in a tree t s.t.

- every guarded set appears in some bag node in t, and
- for each element, the set of bags with this element is connected.

φ has **width** k if the max number of free variables in any subformula is k.

Tree-like models for GFP

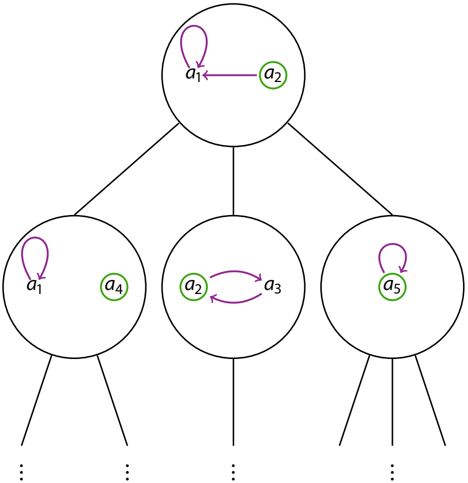
Theorem (Grädel '99)

Every satisfiable $\varphi \in$ GFP of width k has a model of tree width at most $k-1$.

A structure $\mathfrak{A}$ has **tree width** $k-1$ if it can be covered by (overlapping) bags of size at most k, arranged in a tree t s.t.

- every guarded set appears in some bag node in t, and
- for each element, the set of bags with this element is connected.

φ has **width** k if the max number of free variables in any subformula is k.



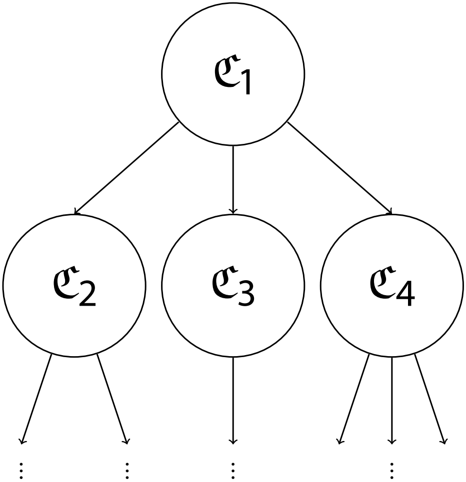
Encoding structures of tree width $k-1$

Fix a set $K = \{a, b, c, \ldots\}$ of names of size $2k$.

Let $\mathbb{K} := \{\mathfrak{C} : \mathfrak{C}$ is a σ-structure with universe $C \subseteq K$ of size at most $k\}$.

A $\mathbb{K}$-**tree** is an unranked infinite tree with

- arbitrary branching (possibly infinite), and
- node labels $\mathfrak{C} \in \mathbb{K}$.



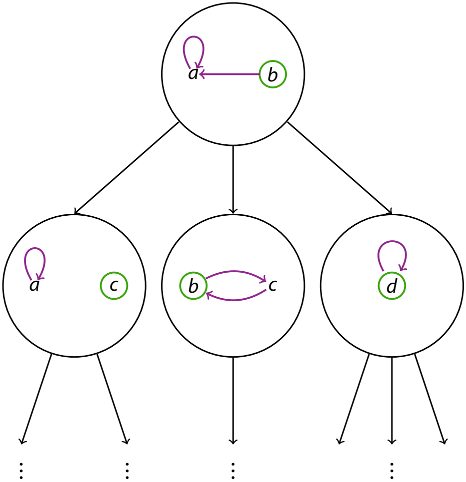
Encoding structures of tree width $k-1$

Fix a set $K = \{a, b, c, \dots\}$ of names of size $2k$.

Let $\mathbb{K} := \{\mathfrak{C} : \mathfrak{C}$ is a σ-structure with universe $C \subseteq K$ of size at most $k\}$.

A $\mathbb{K}$-**tree** is an unranked infinite tree with

- arbitrary branching (possibly infinite), and
- node labels $\mathfrak{C} \in \mathbb{K}$.



Encoding structures of tree width $k - 1$

Fix a set $K = \{a, b, c, \ldots\}$ of names of size $2k$.

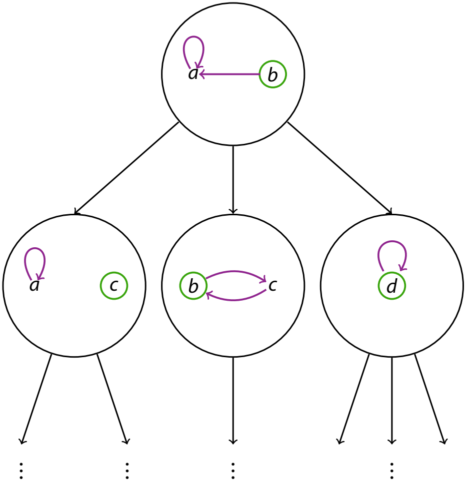
Let $\mathbb{K} := \{\mathfrak{C} : \mathfrak{C}$ is a σ-structure with universe $C \subseteq K$ of size at most $k\}$.

A $\mathbb{K}$-**tree** is an unranked infinite tree with

- arbitrary branching (possibly infinite), and
- node labels $\mathfrak{C} \in \mathbb{K}$.

$\mathbb{K}$-trees are **consistent** if neighboring nodes agree on any shared names.

A consistent $\mathbb{K}$-tree t encodes a σ-structure $\mathfrak{D}(t)$.



Alternating parity automata on infinite unranked trees

$$\mathcal{A} = \langle \mathbb{A}, Q, q_0, \delta, \Omega \rangle$$

δ describes possible moves for Eve and Adam

$\Omega : Q \to P$ for a finite set $P \subseteq \mathbb{N}$ of priorities

Acceptance game $\mathcal{A} \times t$

- Positions in the game are $Q \times \mathrm{dom}(t)$.
- Eve and Adam select the next position in the play based on δ.
- Eve is trying to ensure the play satisfies the **parity condition**: the maximum priority occurring infinitely often in the play is even.

Alternating parity automata on infinite unranked trees

$$\mathcal{A} = \langle \mathbb{A}, Q, q_0, \delta, \Omega \rangle$$

δ describes possible moves for Eve and Adam

$\Omega : Q \to P$ for a finite set $P \subseteq \mathbb{N}$ of priorities

Acceptance game $\mathcal{A} \times t$

- Positions in the game are $Q \times \mathrm{dom}(t)$.
- Eve and Adam select the next position in the play based on δ.
- Eve is trying to ensure the play satisfies the **parity condition**: the maximum priority occurring infinitely often in the play is even.

$L(\mathcal{A}) := \{t : \text{Eve has a winning strategy in } \mathcal{A} \times t\}$

Example

Let $\mathbb{A} := \{\spadesuit, \diamondsuit\}$.

$L := \{t :$ there is some $\spadesuit$ in t s.t.
every downward path from this $\spadesuit$ has infinitely-many $\diamondsuit\}$.

Example

Let $\mathbb{A} := \{\spadesuit, \diamondsuit\}$.

$L := \{t$: there is some $\spadesuit$ in t s.t.
every downward path from this $\spadesuit$ has infinitely-many $\diamondsuit\}$.

Construct $\mathcal{A} := \langle \mathbb{A}, Q, q_0, \delta, \Omega \rangle$ recognizing L with
$Q := \{q_0, r_\spadesuit, r_\diamondsuit\}$ and $\Omega : q_0, r_\spadesuit \mapsto 1; r_\diamondsuit \mapsto 2$.

- In state q_0, Eve chooses a neighbor of the current node.
  If she sees an $\spadesuit$, Eve can choose to switch to state $r_\spadesuit$.
- In state $r_\spadesuit$ or $r_\diamondsuit$ when reading letter $l \in \{\spadesuit, \diamondsuit\}$,
  Adam selects a child in the tree and moves to state r_l.

(Recall that Eve is trying to ensure that the **parity condition** is satisfied:
the maximum priority visited infinitely often is even.)

Automata for GFP

Fix sentence $\varphi \in \mathrm{GFP}[\sigma]$ of width k.

Automata for GFP

Fix sentence $\varphi \in \mathrm{GFP}[\sigma]$ of width k.

Proposition

There is a 1-way parity automaton $\mathcal{C}_{\mathbb{K}}$ that checks if a $\mathbb{K}$-tree is consistent.

There is a 2-way parity automaton $\mathcal{C}_\varphi := \langle \mathbb{K}, Q, q_0, \delta, \Omega \rangle$ that runs on consistent $\mathbb{K}$-trees t and accepts iff φ holds in σ-structure $\mathfrak{D}(t)$.

Automata for GFP

Fix sentence $\varphi \in \mathrm{GFP}[\sigma]$ of width k.

Proposition

There is a 1-way parity automaton $\mathcal{C}_{\mathbb{K}}$ that checks if a $\mathbb{K}$-tree is consistent.

There is a 2-way parity automaton $\mathcal{C}_\varphi := \langle \mathbb{K}, Q, q_0, \delta, \Omega \rangle$ that runs on consistent $\mathbb{K}$-trees t and accepts iff φ holds in σ-structure $\mathfrak{D}(t)$.

State set $Q := \mathrm{cl}(\varphi, K)$ (subformulas of φ with names from K substituted for free vars) and **initial state** $q_0 := \varphi$.

Automata for GFP

Fix sentence $\varphi \in \mathrm{GFP}[\sigma]$ of width k.

Proposition

There is a 1-way parity automaton $\mathcal{C}_{\mathbb{K}}$ that checks if a $\mathbb{K}$-tree is consistent.

There is a 2-way parity automaton $\mathcal{C}_{\varphi} := \langle \mathbb{K}, Q, q_0, \delta, \Omega \rangle$ that runs on consistent $\mathbb{K}$-trees t and accepts iff φ holds in σ-structure $\mathfrak{D}(t)$.

State set $Q := \mathrm{cl}(\varphi, K)$ (subformulas of φ with names from K substituted for free vars) and **initial state** $q_0 := \varphi$.

Transition function δ in state $q \in Q$ at a position labelled $\mathfrak{C}$ with universe C:

- If q is $R\boldsymbol{a}$ or $\neg R\boldsymbol{a}$, then move to $\top$ if $\mathfrak{C} \models q$, and move to $\bot$ otherwise.
- If q is $\psi_1 \vee \psi_2$, then Eve can choose to switch to state ψ_1 or ψ_2.
- If q is $\psi_1 \wedge \psi_2$, then Adam can choose to switch to state ψ_1 or ψ_2.

Automata for GFP

Transition function δ in state $q \in Q$ at a position labelled $\mathfrak{C}$ with universe C

- If q is $\exists \boldsymbol{x}(G(\boldsymbol{ax}) \wedge \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \subseteq C$, then Eve can choose to
  - stay in the same node, choose some $\boldsymbol{b} \subseteq C$ such that $\mathfrak{C} \models G(\boldsymbol{ab})$, and move to state $\psi(\boldsymbol{ab})$, or
  - move to some neighbor (parent or child), and stay in state q.
- If q is $\exists \boldsymbol{x}(G(\boldsymbol{ax}) \wedge \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \nsubseteq C$, then move to state $\bot$.

Automata for GFP

Transition function δ in state $q \in Q$ at a position labelled $\mathfrak{C}$ with universe C

- If q is $\exists \boldsymbol{x}(G(\boldsymbol{ax}) \wedge \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \subseteq C$, then Eve can choose to
  - stay in the same node, choose some $\boldsymbol{b} \subseteq C$ such that $\mathfrak{C} \models G(\boldsymbol{ab})$, and move to state $\psi(\boldsymbol{ab})$, or
  - move to some neighbor (parent or child), and stay in state q.
- If q is $\exists \boldsymbol{x}(G(\boldsymbol{ax}) \wedge \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \nsubseteq C$, then move to state $\bot$.

- If q is $\forall \boldsymbol{x}(G(\boldsymbol{ax}) \rightarrow \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \subseteq C$, then Adam can choose to
  - stay in the same node, choose some $\boldsymbol{b} \subseteq C$ such that $\mathfrak{C} \models G(\boldsymbol{ab})$, and move to state $\psi(\boldsymbol{ab})$, or
  - move to some neighbor (parent or child), and stay in state q.
- If q is $\forall \boldsymbol{x}(G(\boldsymbol{ax}) \rightarrow \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \nsubseteq C$, then move to state $\top$.

Automata for GFP

Transition function δ in state $q \in Q$ at a position labelled $\mathfrak{C}$ with universe C

- If q is $\exists \boldsymbol{x}(G(\boldsymbol{ax}) \wedge \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \subseteq C$, then Eve can choose to
  - stay in the same node, choose some $\boldsymbol{b} \subseteq C$ such that $\mathfrak{C} \models G(\boldsymbol{ab})$, and move to state $\psi(\boldsymbol{ab})$, or
  - move to some neighbor (parent or child), and stay in state q.
- If q is $\exists \boldsymbol{x}(G(\boldsymbol{ax}) \wedge \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \nsubseteq C$, then move to state $\bot$.

- If q is $\forall \boldsymbol{x}(G(\boldsymbol{ax}) \rightarrow \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \subseteq C$, then Adam can choose to
  - stay in the same node, choose some $\boldsymbol{b} \subseteq C$ such that $\mathfrak{C} \models G(\boldsymbol{ab})$, and move to state $\psi(\boldsymbol{ab})$, or
  - move to some neighbor (parent or child), and stay in state q.
- If q is $\forall \boldsymbol{x}(G(\boldsymbol{ax}) \rightarrow \psi(\boldsymbol{ax}))$ and $\boldsymbol{a} \nsubseteq C$, then move to state $\top$.

Assume there is a subformula η of the form $[\mathbf{fp}_{Y,\boldsymbol{y}}.\psi(\boldsymbol{y}, Y, \boldsymbol{Z})](\boldsymbol{x})$.

- If q is $\eta(\boldsymbol{a})$ or $Y\boldsymbol{a}$, then the automaton moves to state $\psi(\boldsymbol{a}, Y, \boldsymbol{Z})$.

Automata for GFP

Ordering $Y_j > \cdots > Y_1$ of fixpoint variables based on nesting
(roughly speaking, outer fixpoint variables appear higher in this ordering).

Priority assignment $\Omega : Q \to \{0, 1 \ldots, 2j\}$

$$\text{fixpoint variable } Y_i \mapsto \begin{cases} 2i - 1 & \text{if } Y_i \text{ corresponds to least fixpoint} \\ 2i & \text{if } Y_i \text{ corresponds to greatest fixpoint} \end{cases}$$

$$\text{existential requirement or } \bot \mapsto 1$$

$$\text{everything else} \mapsto 0$$

Parity condition requires that max priority visited infinitely often is even

$\Rightarrow$ existential requirement is always witnessed and
least fixpoint is only unfolded a finite number of times
(before an outer fixpoint is unfolded).

Complexity of satisfiability for GFP

Theorem
(Grädel, Walukiewicz ’99)

Satisfiability is decidable for GFP in 2EXPTIME (EXPTIME for fixed width).

$$\varphi \in \mathrm{GFP}$$

$$\downarrow$$

2-way parity automaton

$$\mathcal{A}_\varphi := \mathcal{C}_\varphi \wedge \mathcal{C}_\mathbb{K}$$

$$\downarrow$$

$$L(\mathcal{A}_\varphi) \neq \varnothing?$$

Complexity of satisfiability for GFP

Theorem
(Grädel, Walukiewicz ’99)

Satisfiability is decidable for GFP in 2EXPTIME (EXPTIME for fixed width).

$\varphi \in \text{GFP}$

$\downarrow$

2-way parity automaton

$\mathcal{A}_\varphi := \mathcal{C}_\varphi \wedge \mathcal{C}_\mathbb{K}$

$\downarrow$ EXPTIME using [Vardi’98]

$L(\mathcal{A}_\varphi) \neq \varnothing$?

Complexity of satisfiability for GFP

Theorem
(Grädel, Walukiewicz '99)

Satisfiability is decidable for GFP in 2EXPTIME (EXPTIME for fixed width).

Similar techniques yield 2EXPTIME complexity for GNFP satisfiability testing.

$\varphi \in \mathrm{GFP}$

$\downarrow$

2-way parity automaton

$\mathcal{A}_\varphi := \mathcal{C}_\varphi \wedge \mathcal{C}_\mathbb{K}$

$\downarrow$ EXPTIME using [Vardi'98]

$L(\mathcal{A}_\varphi) \neq \varnothing$?

Summary

Tree automata are a useful tool to decide **satisfiability** for expressive logics like GFP and GNFP that have tree-like models.

Summary

Tree automata are a useful tool to decide **satisfiability** for expressive logics like GFP and GNFP that have tree-like models.

But we can do more...

Boundedness

Let $\psi(\boldsymbol{y}, Y)$ positive in Y.

For all $\mathfrak{A}$, ψ induces a monotone operation $V \mapsto \psi_{\mathfrak{A}}(V) := \{\boldsymbol{a} : \mathfrak{A}, \boldsymbol{a}, V \models \psi\}$
$\Rightarrow$ there is a unique least fixpoint $\bigcup_\alpha \psi_{\mathfrak{A}}^\alpha$.

$$\psi_{\mathfrak{A}}^0 := \emptyset$$
$$\psi_{\mathfrak{A}}^{\alpha+1} := \psi_{\mathfrak{A}}(\psi_{\mathfrak{A}}^\alpha)$$
$$\psi_{\mathfrak{A}}^\lambda := \bigcup_{\alpha<\lambda} \psi_{\mathfrak{A}}^\alpha$$

Boundedness

Let $\psi(\boldsymbol{y}, Y)$ positive in Y.

For all $\mathfrak{A}$, ψ induces a monotone operation $V \mapsto \psi_{\mathfrak{A}}(V) := \{\boldsymbol{a} : \mathfrak{A}, \boldsymbol{a}, V \models \psi\}$
$\Rightarrow$ there is a unique least fixpoint $\bigcup_\alpha \psi^\alpha_{\mathfrak{A}}$.

$$\psi^0_{\mathfrak{A}} := \varnothing$$
$$\psi^{\alpha+1}_{\mathfrak{A}} := \psi_{\mathfrak{A}}(\psi^\alpha_{\mathfrak{A}})$$
$$\psi^\lambda_{\mathfrak{A}} := \bigcup_{\alpha<\lambda} \psi^\alpha_{\mathfrak{A}}$$

Boundedness problem for $\mathcal{L}$

Input: $\psi(\boldsymbol{y}, Y) \in \mathcal{L}$ positive in Y

Question: is there $n \in \mathbb{N}$ s.t. for all structures $\mathfrak{A}$, $\psi^n_{\mathfrak{A}} = \psi^{n+1}_{\mathfrak{A}}$?
(i.e. the least fixpoint is always reached within n iterations)

Boundedness

Proposition

For ψ in GFP or GNFP of width k, ψ is bounded over all structures
iff ψ is bounded over tree-like structures (of tree width $k - 1$).

Boundedness

Proposition

For ψ in GFP or GNFP of width k, ψ is bounded over all structures iff ψ is bounded over tree-like structures (of tree width $k-1$).

$\Rightarrow$ boundedness amenable to techniques using tree automata

Boundedness

Proposition

For ψ in GFP or GNFP of width k, ψ is bounded over all structures iff ψ is bounded over tree-like structures (of tree width $k-1$).

$\Rightarrow$ boundedness amenable to techniques using tree automata

Construct 2-way parity automaton $\mathcal{A}_\varphi$ for $\varphi := [\mathbf{lfp}_{X,\boldsymbol{x}}.\psi(\boldsymbol{x}, X)](\boldsymbol{x})$ as before.

Add a counter which is incremented each time the least fixpoint is unfolded (and is untouched otherwise).

This new automaton $\mathcal{B}_\varphi$ is a **cost automaton.**
Boundedness of ψ is related to boundedness of function defined by $\mathcal{B}_\varphi$.

Cost automata on infinite trees

$$\mathcal{B} = \langle \mathbb{A}, Q, q_0, \delta, \Omega \rangle$$

δ describes possible moves for Eve and Adam, and associated counter actions (increment, reset, leave unchanged)

$\Omega : Q \to P$ for a finite set $P \subseteq \mathbb{N}$ of priorities

***n*-acceptance game** $\mathcal{B} \times t$

- Positions in the game are $Q \times \mathrm{dom}(t)$.
- Eve and Adam select the next position in the play based on δ.
- Eve is trying to ensure the play has counter value at most n and the maximum priority occurring infinitely often in the play is even.

Semantics $[\![\mathcal{B}]\!] : \mathbb{A}\text{-trees} \to \mathbb{N} \cup \{\infty\}$

$[\![\mathcal{B}]\!](t) := \inf \{n : \text{Eve wins the } n\text{-acceptance game } \mathcal{B} \times t\}$

Boundedness for cost automata

Boundedness problem for cost automata

Input: cost automaton $\mathcal{B}$

Question: is there $n \in \mathbb{N}$ such that for all trees t, $[\![\mathcal{B}]\!](t) \leq n$?

Boundedness for cost automata

Boundedness problem for cost automata

Input: cost automaton $\mathcal{B}$

Question: is there $n \in \mathbb{N}$ such that for all trees t, $[\![\mathcal{B}]\!](t) \leq n$?

Decidability of boundedness is not known in general for cost automata over infinite trees...

Boundedness for cost automata

Boundedness problem for cost automata

Input: cost automaton $\mathcal{B}$

Question: is there $n \in \mathbb{N}$ such that for all trees t, $[\![\mathcal{B}]\!](t) \leq n$?

Decidability of boundedness is not known in general for cost automata over infinite trees...

...but we are interested in special cases using **distance-parity automata**:
1 counter that is only incremented or left unchanged (never reset)
for which boundedness is known to be decidable.

Complexity of boundedness for guarded logics

Theorem
(Benedikt, Colcombet, ten Cate, VB. '15)

Boundedness for GFP is decidable in elementary time.

$\psi \in \mathrm{GFP}$

↓

2-way distance-parity automaton $\mathcal{B}$

↓

$[\![\mathcal{B}]\!]$ bounded?

Complexity of boundedness for guarded logics

Theorem
(Benedikt, Colcombet, ten Cate, VB. ’15)

Boundedness for GFP is decidable in elementary time.

$\psi \in \mathrm{GFP}$

↓

2-way distance-parity automaton $\mathcal{B}$

↓

$[\![\mathcal{B}]\!]$ bounded?

Theorem
(Benedikt, Colcombet, ten Cate, VB. '15)

Boundedness for GFP is decidable in elementary time.

Similar techniques yield elementary complexity for GNFP boundedness.

Improves upon results of [Blumensath, Otto, Weyer '14], [Bárány, ten Cate, Otto '12].

$\psi \in \mathrm{GFP}$

↓

2-way distance-parity automaton $\mathcal{B}$

↓

$[\![\mathcal{B}]\!]$ bounded?

Summary

Tree automata are a useful tool to decide **satisfiability** for expressive logics like GFP and GNFP that have tree-like models.

Cost automata can be used to decide **boundedness** for these logics (Benedikt, Colcombet, ten Cate, VB. ’15)

Summary

Tree automata are a useful tool to decide **satisfiability** for expressive logics like GFP and GNFP that have tree-like models.

Cost automata can be used to decide **boundedness** for these logics (Benedikt, Colcombet, ten Cate, VB. '15)

Automata used to prove **uniform interpolation** for L_{μ}, and this automata-logic connection can be used to prove interpolation for UNFP (Benedikt, ten Cate, VB. '15)

Automata for GNFP

Proposition

For all $\psi(\boldsymbol{a}) \in \mathrm{cl}(\varphi, K)$, there is a 2-way localized parity automaton $\mathcal{A}^{\ell}_{\psi(\boldsymbol{a})}$ running on $\mathbb{K}$-trees t such that

$$\mathcal{A}^{\ell}_{\psi(\boldsymbol{a})} \text{ accepts } t \text{ starting from } v \quad \text{iff} \quad \mathfrak{D}(t), [v, a_1], \ldots, [v, a_j] \models \psi(\boldsymbol{x}).$$

Construct inductively. In general, on input t:

- Eve guesses an annotation t' of t with subformulas from $\mathrm{cl}(\varphi, K)$ and checks $\psi(\boldsymbol{a})$ on t' (assuming annotations are correct),
- Adam can challenge some $\eta(\boldsymbol{a}')$ in the annotation by launching (inductively defined) $\mathcal{A}^{\ell}_{\eta(\boldsymbol{a}')}$.

Complexity of satisfiability for GNFP

Theorem
(Bárány, ten Cate, Segoufin '11)

Satisfiability is decidable for GNFP in 2EXPTIME (even for fixed width).

Automata approach:
Benedikt, Colcombet, ten Cate, VB. '15

$\varphi \in \text{GNFP}$

$\downarrow$

2-way localized automata $\mathcal{A}^{\ell}_{\psi(\boldsymbol{a})}$

$\downarrow$

2-way parity automaton

$$\mathcal{A}_\varphi \;:=\; \mathcal{A}^{\ell}_{\varphi} \wedge \mathcal{C}_{\mathbb{K}}$$

$\downarrow$

$L(\mathcal{A}_\varphi) \neq \emptyset$?