Branching-Time Model Checking Gap-Order Constraint Systems

Richard Mayr Patrick Totzke

University of Edinburgh, UK

September 26, 2013

Gap Clauses

Def: Gap Clauses

$$x - y \geq k$$

where x, y are integer variables or constants and $k \in \mathbb{Z}$.

Gap Clauses/Constraints

Def: Gap Constraints

$$\bigwedge_{0 \leq i \leq n} (x_i - y_i \geq k_i)$$

where x_i, y_i are integer variables or constants and $k_i \in \mathbb{Z}$.

Gap Clauses/Constraints

Def: *positive* Gap Constraints

$$\bigwedge_{0 \leq i \leq n} (x_i - y_i \geq k_i)$$

where x_i, y_i are integer variables or constants and $k_i \in \mathbb{N}$.

Gap Clauses/Constraints

Def: *positive* Gap Constraints

$$\bigwedge_{0 \leq i \leq n} (x_i - y_i \geq k_i)$$

where x_i, y_i are integer variables or constants and $k_i \in \mathbb{N}$.

positive GC are not negation-closed!

Gap Clauses/Constraints

Def: *positive* Gap Constraints

$$\bigwedge_{0 \leq i \leq n} (x_i - y_i \geq k_i)$$

where x_i, y_i are integer variables or constants and $k_i \in \mathbb{N}$.

positive GC are not negation-closed!

Write

$$\begin{aligned} \mathit{Var} &= \{x, y, \ldots\} \text{ for the } \mathit{variables} \\ \mathit{Const} &\subset \mathbb{Z} \text{ for the } \mathit{constants} \text{ and} \\ \mathit{Val} &\text{ for the set of } \mathit{valuations}\ \nu : \mathit{Var} \to \mathbb{Z}. \end{aligned}$$

Gap Constraints

1. can characterise subsets $S \subseteq Val$ (of satisfied valuations)
2. can determine how valuations evolve:

Gap Constraints

1. can characterise subsets $S \subseteq \mathit{Val}$ (of satisfied valuations)
2. can determine how valuations evolve: For instance,

$$x - x' \geq 0$$

means the value of x does not increase.

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Step semantics:

$$\nu \longrightarrow \nu' \text{ iff } \nu \oplus \nu' \models \mathcal{C} \text{ for some } \mathcal{C} \in \Delta.$$

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Step semantics:

$$\nu \longrightarrow \nu' \text{ iff } \nu \oplus \nu' \models \mathcal{C} \text{ for some } \mathcal{C} \in \Delta.$$

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Step semantics:

$$\nu \longrightarrow \nu' \text{ iff } \nu \oplus \nu' \models \mathcal{C} \text{ for some } \mathcal{C} \in \Delta.$$

$$\nu \oplus \nu'(x) = \begin{cases} \nu(x), & \text{if } x \in \mathit{Var} \\ \nu'(x), & \text{if } x \in \mathit{Var}'. \end{cases}$$

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Step semantics:

$$\nu \longrightarrow \nu' \text{ iff } \nu \oplus \nu' \models \mathcal{C} \text{ for some } \mathcal{C} \in \Delta.$$

Example:

$\mathcal{C}_1 = (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)$
$\mathcal{C}_2 = (y - y' \geq 1) \wedge (x' - x \geq 0) \wedge (y' - 0 \geq 0)$

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Step semantics:

$$\nu \longrightarrow \nu' \text{ iff } \nu \oplus \nu' \models \mathcal{C} \text{ for some } \mathcal{C} \in \Delta.$$

Example:

$\mathcal{C}_1 = (x > x' \geq 0) \wedge (y' = y)$
$\mathcal{C}_2 = (x \leq x') \wedge (y > y' \geq 0)$

Gap-Order Constraint Systems

Definition (CGS)

are given by finite sets

Var of variables ranging over $\mathbb{Z}$,

Const of integer constants, and

Δ of *positive* transitional gap contraints.

Step semantics:

$$\nu \longrightarrow \nu' \text{ iff } \nu \oplus \nu' \models \mathcal{C} \text{ for some } \mathcal{C} \in \Delta.$$

Example: lex. Countdown of (y, x)

$\mathcal{C}_1 = (x > x' \geq 0) \wedge (y' = y)$
$\mathcal{C}_2 = (x \leq x') \wedge (y > y' \geq 0)$

Overapproximating Counter Machines

Zero-tests

$(c_1 - 0 \geq 0) \wedge (0 - c_1 \geq 0)$

Overapproximating Counter Machines

Zero-tests

$(c_1 = 0)$

Overapproximating Counter Machines

Zero-tests

$(c_1 = 0)$

Finite Control

$(state = 0) \wedge (state' = 1) \qquad // \ s_0 \longrightarrow s_1$

Overapproximating Counter Machines

Zero-tests

$(c_1 = 0)$

Finite Control

$(state = 0) \wedge (state' = 1)$ // $s_0 \longrightarrow s_1$

Increments/Decrements

$(c_1' - c_1 \geq 0)$ //$c_1{++}$

Overapproximating Counter Machines

Zero-tests

$(c_1 = 0)$

Finite Control

$(state = 0) \wedge (state' = 1)$ $\quad // \; s_0 \longrightarrow s_1$

Increments/Decrements

$(c_1' - c_1 \geq 0)$ $\quad //c_1$++

$(c_1 - c_1' \geq 0) \wedge (c_1' - 0 \geq 0)$ $\quad //c_1$- -

Overapproximating Counter Machines

Zero-tests

$(c_1 = 0)$

Finite Control

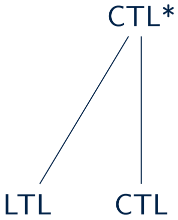
$(state = 0) \wedge (state' = 1)$ $\quad // \; s_0 \longrightarrow s_1$

Increments/Decrements **are imprecise!**

$(c_1' - c_1 \geq 0)$ $\quad // c_1{++}$

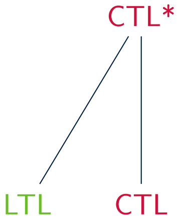
$(c_1 - c_1' \geq 0) \wedge (c_1' - 0 \geq 0)$ $\quad // c_1$- -

Model Checking GCS



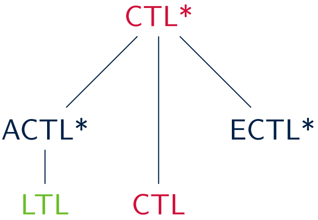
[Čer94] Model checking CTL undecidable, but LTL is decidable for IRA.

Model Checking GCS



[Čer94] Model checking CTL undecidable, but LTL is decidable for IRA.

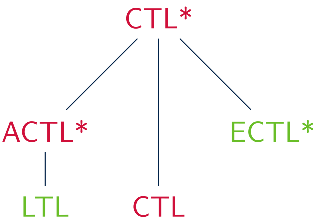
Model Checking GCS



[Čer94] Model checking CTL undecidable, but LTL is decidable for IRA.

[BP12] LTL and ECTL* are *PSPACE*-complete; ACTL* is undecidable for GCS.

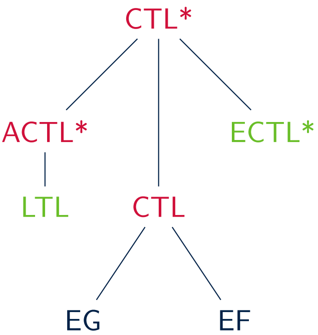
Model Checking GCS



[Čer94] Model checking CTL undecidable, but LTL is decidable for IRA.

[BP12] LTL and ECTL* are *PSPACE*-complete; ACTL* is undecidable for GCS.

Model Checking GCS

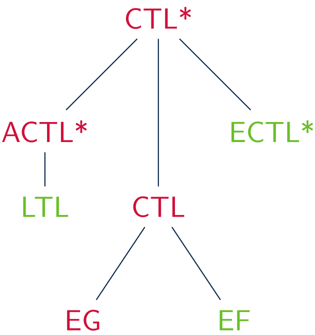


[Čer94] Model checking CTL undecidable, but LTL is decidable for IRA.

[BP12] LTL and ECTL* are *PSPACE*-complete; ACTL* is undecidable for GCS.

We EF is decidable and EG undecidable for GCS.

Model Checking GCS



[Čer94] Model checking CTL undecidable, but LTL is decidable for IRA.

[BP12] LTL and ECTL* are *PSPACE*-complete; ACTL* is undecidable for GCS.

We EF is decidable and EG undecidable for GCS.

CTL over Gap Clauses

Syntax

$$\psi ::= \mathcal{C} \mid \neg\psi \mid \psi \vee \psi \mid X\psi \mid EF\psi \mid EG\psi \mid (\psi U \psi)$$

EG over Gap Clauses

Syntax

$$\psi ::= \mathcal{C} \mid \neg\psi \mid \psi \vee \psi \mid \mathit{X}\psi \mid \cancel{\mathit{EF}\psi \mid} \quad \mathit{EG}\psi \quad \cancel{\mid (\psi \mathit{U} \psi)}$$

EG over Gap Clauses

Syntax

$$\psi ::= \mathcal{C} \mid \neg\psi \mid \psi \vee \psi \mid X\psi \mid \cancel{EF\psi \mid} \quad EG\psi \quad \cancel{\mid (\psi U \psi)}$$

- EG model checking GCS is undecidable.
- Proof by enforcing exact increments/decrements (Simulating Minski machines).

EF over Gap Clauses

Syntax

$$\psi ::= \mathcal{C} \mid \neg\psi \mid \psi \vee \psi \mid X\psi \mid EF\psi \mid \cancel{EG\psi} \quad \cancel{\mid (\psi U \psi)}$$

EF over Gap Clauses

Syntax

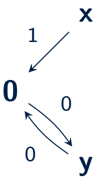
$$\psi ::= \mathcal{C} \mid \neg\psi \mid \psi \vee \psi \mid X\psi \mid EF\psi \mid \cancel{EG\psi} \quad \cancel{\mid (\psi U \psi)}$$

- EF model checking GCS is decidable.
- Proof by finding finite representation for $Sat(\mathcal{C})$ that is closed under negation, union, Pre and Pre*.

Monotonicity Graphs

Gap Constraints as finite labeled graphs over *Var* $\cup$ *Const*

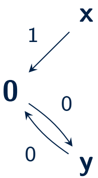
$$\begin{aligned}\mathcal{C} = & (x - 0 \geq 1) \\ & \wedge (y - 0 \geq 0) \\ & \wedge (0 - y \geq 0)\end{aligned} \quad \sim$$



Monotonicity Graphs

Gap Constraints as finite labeled graphs over *Var* ∪ *Const*

$$\begin{aligned} \mathcal{C} = &(x - 0 \geq 1) \\ &\wedge (y - 0 \geq 0) \\ &\wedge (0 - y \geq 0) \end{aligned} \quad \sim$$

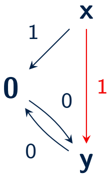


- *Degree* of MG: inverse of minimal negative value

Monotonicity Graphs

Gap Constraints as finite labeled graphs over *Var* ∪ *Const*

$$
\begin{aligned}
\mathcal{C} = & (x - 0 \geq 1) \\
& \wedge (y - 0 \geq 0) \\
& \wedge (0 - y \geq 0)
\end{aligned}
\quad \sim
$$

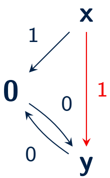


- *Degree* of MG: inverse of minimal negative value
- *Closure* of MG has same denotation

Monotonicity Graphs

Gap Constraints as finite labeled graphs over *Var* ∪ *Const*

$$
\begin{aligned}
\mathcal{C} = &(x - 0 \geq 1)\\
&\wedge (y - 0 \geq 0)\\
&\wedge (0 - y \geq 0)
\end{aligned}
\qquad \sim
$$

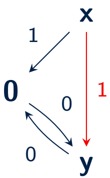


- *Degree* of MG: inverse of minimal negative value
- *Closure* of MG has same denotation
- represent $S \subseteq Val$ by finite sets of (arbitrary) MG.

Monotonicity Graphs

Gap Constraints as finite labeled graphs over *Var* $\cup$ *Const*

$$\begin{aligned}\mathcal{C} = &(x - 0 \geq 1)\\ &\wedge(y - 0 \geq 0)\\ &\wedge(0 - y \geq 0)\end{aligned} \quad \sim$$

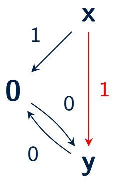


- *Degree* of MG: inverse of minimal negative value
- *Closure* of MG has same denotation
- represent $S \subseteq \mathit{Val}$ by finite sets of (arbitrary) MG. Example: $\{M_{\mathcal{C}}\}$ represents $S = \mathit{Sat}(\mathcal{C}) = \{\nu \mid \nu(x) > \nu(y) = 0\}$.

Monotonicity Graphs

Gap Constraints as finite labeled graphs over *Var* $\cup$ *Const*

$$\mathcal{C} = (x - 0 \geq 1) \wedge (y - 0 \geq 0) \wedge (0 - y \geq 0) \quad \sim$$



- *Degree* of MG: inverse of minimal negative value
- *Closure* of MG has same denotation
- represent $S \subseteq Val$ by finite sets of (arbitrary) MG. Example: $\{M_{\mathcal{C}}\}$ represents $S = Sat(\mathcal{C}) = \{\nu \mid \nu(x) > \nu(y) = 0\}$.

$$\psi ::= \mathcal{C} \mid \psi \vee \varphi \mid \neg\psi \mid X\psi \mid EF\psi$$

Negation

$$Rep(S) = \quad \{M_0, M_1, \ldots, M_k\}$$

Negation

$$Rep(S) = \quad \{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\mathcal{C}_{M_0} \vee \mathcal{C}_{M_1} \vee \cdots \vee \mathcal{C}_{M_k}$$

Negation

$$Rep(S) = \quad \{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\mathcal{C}_{M_0} \vee \mathcal{C}_{M_1} \vee \cdots \vee \mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

Negation

$$\neg Rep(S) = \neg\{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\neg\mathcal{C}_{M_0} \wedge \neg\mathcal{C}_{M_1} \wedge \cdots \wedge \neg\mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

$\rightsquigarrow$ propagate negations to clauses

Negation

$$\neg Rep(S) = \neg\{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\neg\mathcal{C}_{M_0} \wedge \neg\mathcal{C}_{M_1} \wedge \cdots \wedge \neg\mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

- $\leadsto$ propagate negations to clauses
- $\leadsto$ negate clauses

Negation

$$\neg Rep(S) = \neg\{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\neg\mathcal{C}_{M_0} \wedge \neg\mathcal{C}_{M_1} \wedge \cdots \wedge \neg\mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

$\rightsquigarrow$ propagate negations to clauses

$\rightsquigarrow$ negate clauses

$\rightsquigarrow$ bring to DNF

Negation

$$\neg Rep(S) = \neg\{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\neg\mathcal{C}_{M_0} \wedge \neg\mathcal{C}_{M_1} \wedge \cdots \wedge \neg\mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

- $\rightsquigarrow$ propagate negations to clauses
- $\rightsquigarrow$ negate clauses
- $\rightsquigarrow$ bring to DNF
- $\rightsquigarrow$ interpret as set of MG

Negation

$$\neg Rep(S) = \neg\{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\neg\mathcal{C}_{M_0} \wedge \neg\mathcal{C}_{M_1} \wedge \cdots \wedge \neg\mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

- $\rightsquigarrow$ propagate negations to clauses
- $\rightsquigarrow$ negate clauses $\leftarrow$ increases degree
- $\rightsquigarrow$ bring to DNF
- $\rightsquigarrow$ interpret as set of MG

Negation

$$\neg Rep(S) = \neg\{M_0, M_1, \ldots, M_k\}$$

$$\sim$$

$$\neg\mathcal{C}_{M_0} \wedge \neg\mathcal{C}_{M_1} \wedge \cdots \wedge \neg\mathcal{C}_{M_k}$$

... is a Gap-Formula in DNF.

$\rightsquigarrow$ propagate negations to clauses

$\rightsquigarrow$ negate clauses $\leftarrow$ increases degree

$$x - y \not\geq k \iff y - x \geq -(k-1)$$

$\rightsquigarrow$ bring to DNF

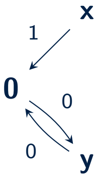
$\rightsquigarrow$ interpret as set of MG

Computing *Pre*

$$S = \{\nu \mid \nu(x) > \nu(y) = 0\}$$

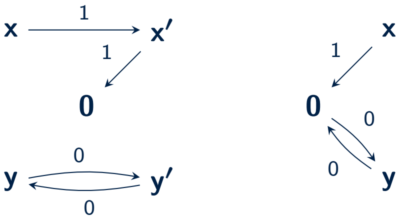
Computing *Pre*

$S \ = \{\nu \mid \nu(x) > \nu(y) = 0\}$



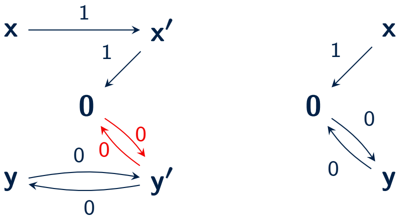
Computing *Pre*

$$
\begin{aligned}
S &= \{\nu \mid \nu(x) > \nu(y) = 0\} \\
\mathcal{C}_1 &= (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)
\end{aligned}
$$



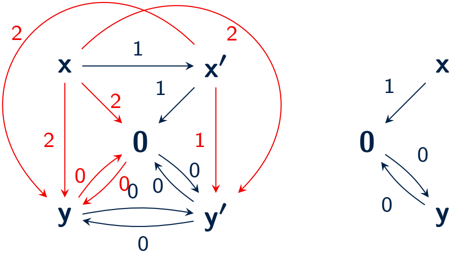
Computing *Pre*

$$S \ = \{\nu \mid \nu(x) > \nu(y) = 0\}$$
$$\mathcal{C}_1 = (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)$$



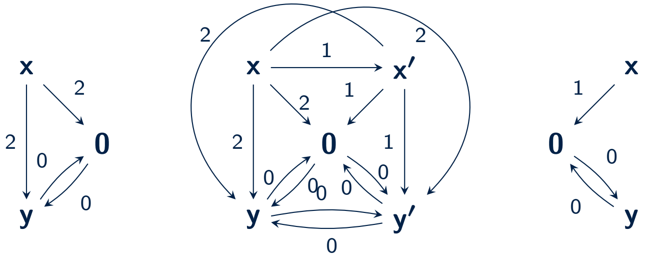
Computing *Pre*

$$S \;= \{\nu \mid \nu(x) > \nu(y) = 0\}$$
$$\mathcal{C}_1 = (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)$$



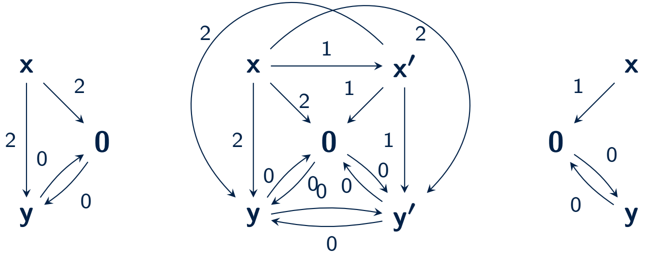
Computing *Pre*

$$S = \{\nu \mid \nu(x) > \nu(y) = 0\}$$
$$\mathcal{C}_1 = (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)$$



Computing *Pre*

$$S\ = \{\nu \mid \nu(x) > \nu(y) = 0\}$$
$$\mathcal{C}_1 = (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)$$

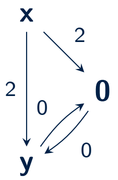


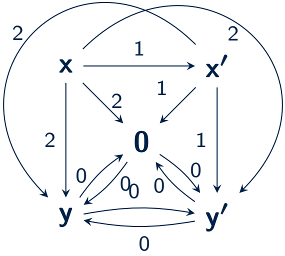
$Pre(C_1, S)$

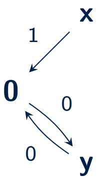
S

Computing *Pre*

$$S = \{\nu \mid \nu(x) > \nu(y) = 0\}$$
$$\mathcal{C}_1 = (x - x' \geq 1) \wedge (y' - y \geq 0) \wedge (y - y' \geq 0) \wedge (x' - 0 \geq 0)$$







$Pre(\mathcal{C}_1, S)$

S

NB: Degree does not increase

Computing Pre^*

Definition ($\sqsubseteq$)

$M \sqsubseteq M'$ if $M(x, y) \leq M'(x, y)$ for all $x, y \in \mathit{Var} \cup \mathit{Const}$.

Computing Pre^*

Definition ($\sqsubseteq$)

$M \sqsubseteq M'$ if $M(x, y) \leq M'(x, y)$ for all $x, y \in \mathit{Var} \cup \mathit{Const}$.

1. $M \sqsubseteq M'$ implies $[\![M]\!] \supseteq [\![M']\!]$

Computing Pre^*

Definition ($\sqsubseteq$)

$M \sqsubseteq M'$ if $M(x, y) \leq M'(x, y)$ for all $x, y \in Var \cup Const$.

1. $M \sqsubseteq M'$ implies $[\![M]\!] \supseteq [\![M']\!]$
2. $\sqsubseteq$ is a well-order over MG^n

Computing Pre^*

Definition ($\sqsubseteq$)

$M \sqsubseteq M'$ if $M(x,y) \leq M'(x,y)$ for all $x, y \in Var \cup Const$.

1. $M \sqsubseteq M'$ implies $[\![M]\!] \supseteq [\![M']\!]$
2. $\sqsubseteq$ is a well-order over MG^n

Compute $Pre^*(M)$:

iteratively unfold the finite! backwards coverability tree and take the union of all nodes...

EF Model Checking GCS is decidable

Theorem

For given GCS and EF formula φ, the set $Sat(\varphi)$ is effectively Gap-definable.

EF Model Checking GCS is decidable

Theorem

For given GCS and EF formula φ, the set $Sat(\varphi)$ is effectively Gap-definable.

Works even with

- arbitrary gap-formulae as atoms and
- positive (trans.) gap-constraints on X/EF operators.

WIP: Equivalence Checking

1. Bisimulation
   - $GCS \approx FS$ is decidable using char. formulae in EF
   - Strong Bisimulation $GCS \sim GCS$ is undecidable

WIP: Equivalence Checking

1. Bisimulation
   - $GCS \approx FS$ is decidable using char. formulae in EF
   - Strong Bisimulation $GCS \sim GCS$ is undecidable
2. Trace inclusion/equivalence
   - $GCS \subseteq GCS$ is in EXPSPACE
   - Universality is EXPSPACE-hard

WIP: Equivalence Checking

1. Bisimulation
   - $GCS \approx FS$ is decidable using char. formulae in EF
   - Strong Bisimulation $GCS \sim GCS$ is undecidable
2. Trace inclusion/equivalence
   - $GCS \subseteq GCS$ is in EXPSPACE
   - Universality is EXPSPACE-hard
3. Simulation Preorder
   - $GCS \preceq FS$ and vv. are decidable (wqo)
   - $GCS \preceq GCS$? WIP.

References

P. A. Abdulla and G. Delzanno. "Constrained Multiset Rewriting". In: *Proc. AVIS'06, 5th int. workshop on on Automated Verification of InfiniteState Systems*. 2006.

L. Bozzelli. "Strong Termination for Gap-Order Constraint Abstractions of Counter Systems". In: *LATA*. 2012, pp. 155–168.

L. Bozzelli and S. Pinchinat. "Verification of Gap-Order Constraint Abstractions of Counter Systems". In: *VMCAI*. 2012, pp. 88–103.

K. Čerāns. "Deciding Properties of Integral Relational Automata". In: *ICALP*. 1994, pp. 35–46.

L. Fribourg and J. Richardson. "Symbolic Verification with Gap-Order Constraints". In: *LOPSTR*. 1996, pp. 20–37.

L. Segoufin and S. Torunczyk. "Automata based verification over linearly ordered data domains ". In: *STACS*. Vol. 9. Dagstuhl, Germany, 2011, pp. 81–92.

References

P. A. Abdulla and G. Delzanno. “Constrained Multiset Rewriting”. In: *Proc. AVIS’06, 5th int. workshop on on Automated Verification of InfiniteState Systems*. 2006.

L. Bozzelli. “Strong Termination for Gap-Order Constraint Abstractions of Counter Systems”. In: *LATA*. 2012, pp. 155–168.

L. Bozzelli and S. Pinchinat. “Verification of Gap-Order Constraint Abstractions of Counter Systems”. In: *VMCAI*. 2012, pp. 88–103.

K. Čerāns. “Deciding Properties of Integral Relational Automata”. In: *ICALP*. 1994, pp. 35–46.

L. Fribourg and J. Richardson. “Symbolic Verification with Gap-Order Constraints”. In: *LOPSTR*. 1996, pp. 20–37.

L. Segoufin and S. Torunczyk. “Automata based verification over linearly ordered data domains ”. In: *STACS*. Vol. 9. Dagstuhl, Germany, 2011, pp. 81–92.

Questions?