Nested Antichains for WS1S

Tomáš Fiedor[1,2] Lukáš Holík[2]

[1]Red Hat, Czech Republic

Ondřej Lengál[2] Tomáš Vojnar[2]

[2]Brno University of Technology, Czech Republic

AVM'15

WS1S

- weak monadic second-order logic of one successor
  - second-order $\Rightarrow$ quantification over relations;
  - monadic $\Rightarrow$ relations are unary (i.e. sets);
  - weak $\Rightarrow$ sets are finite;
  - of one successor $\Rightarrow$ reasoning about linear structures.
- corresponds to finite automata [Büchi'60]
- decidable

WS1S

- weak monadic second-order logic of one successor
  - second-order $\Rightarrow$ quantification over relations;
  - monadic $\Rightarrow$ relations are unary (i.e. sets);
  - weak $\Rightarrow$ sets are finite;
  - of one successor $\Rightarrow$ reasoning about linear structures.
- corresponds to finite automata [Büchi'60]
- decidable — but **NONELEMENTARY**
  - constructive proof via translation to finite automata

Application of WS1S

- allows one to define rich invariants
- famous decision procedure: the MONA tool
  - often efficient (in practice)
- used in tools for checking structural invariants
  - Pointer Assertion Logic Engine (`PALE`)
  - STRucture ANd Data (`STRAND`)
- many other applications
  - program and protocol verifications, linguistics, theorem provers …

Application of WS1S

- allows one to define rich invariants
- famous decision procedure: the MONA tool
  - often efficient (in practice)
- used in tools for checking structural invariants
  - Pointer Assertion Logic Engine (`PALE`)
  - STRucture ANd Data (`STRAND`)
- many other applications
  - program and protocol verifications, linguistics, theorem provers ...
- but sometimes the complexity strikes back
  - unavoidable in general
  - however, we try to push the usability border further
    - using the recent advancements in non-deterministic automata

WS1S

- Syntax:
  - term ψ ::= $X \subseteq Y \mid \mathrm{Sing}(X) \mid X = \{0\} \mid X = \sigma(Y)$
  - formula φ ::= $\psi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists X.\varphi$

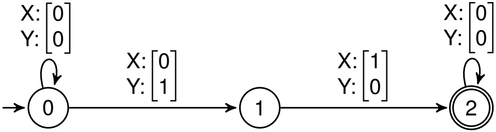
- Interpretation: over finite subsets of $\mathbb{N}$
  - models of formulae = assignments of sets to variables
- sets can be encoded as binary strings:
  - $\{1, 4, 5\} \rightarrow$

Index:	012345		0123456		01234567
Membership:	x✓xx✓✓	,	x✓xx✓✓x	or	x✓xx✓✓xx	…
Encoding:	010011		0100110		01001100

- for each variable we have one track in the alphabet
  - e.g. $\begin{bmatrix}0\\0\end{bmatrix}$ is symbol
- Example: $\{X_1 \mapsto \emptyset, X_2 \mapsto \{4, 2\}\} \models \varphi \overset{def}{\Leftrightarrow} \begin{array}{l} X_1: \\ X_2: \end{array} \begin{bmatrix}0\\0\end{bmatrix} \begin{bmatrix}0\\0\end{bmatrix} \begin{bmatrix}0\\1\end{bmatrix} \begin{bmatrix}0\\0\end{bmatrix} \begin{bmatrix}0\\1\end{bmatrix} \in L(\mathcal{A}_\varphi)$

Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$

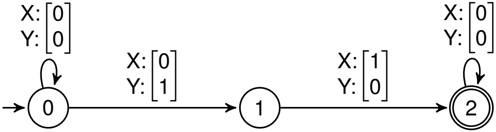


- Example:

$$\neg(X \subseteq Y) \wedge \exists Z.\,\mathrm{Sing}(Z) \vee \exists W.\, W = \sigma(Z)$$

Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$

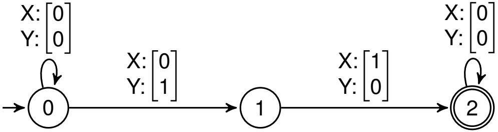


- Example:

$$\neg \underbrace{(X \subseteq Y)}_{\mathcal{A}_3} \wedge \exists Z.\, \underbrace{\mathrm{Sing}(Z)}_{\mathcal{A}_2} \vee \exists W.\, \underbrace{W = \sigma(Z)}_{\mathcal{A}_1}$$

Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$



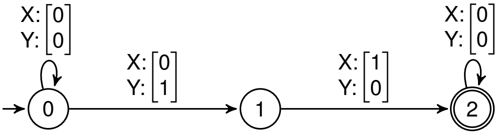
- Example:

$$\neg(\underset{\mathcal{A}_3}{X \subseteq Y}) \land \exists Z.\,\underset{\mathcal{A}_2}{\mathrm{Sing}(Z)} \lor \exists W.\, \underset{\mathcal{A}_1}{W = \sigma(Z)}$$

project W $\longrightarrow \mathcal{A}_4$

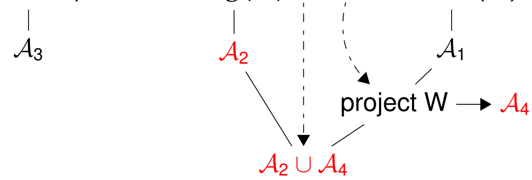
Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$



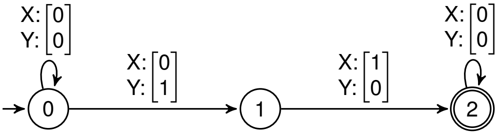
- Example:

$$\neg(X \subseteq Y) \wedge \exists Z.\, \mathrm{Sing}(Z) \vee \exists W.\, W = \sigma(Z)$$



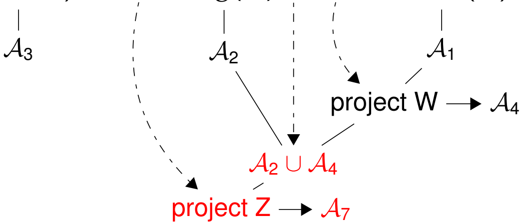
Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$



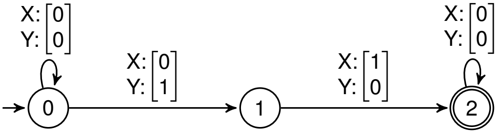
- Example:

$$\neg(X \subseteq Y) \wedge \exists Z.\, \mathrm{Sing}(Z) \vee \exists W.\, W = \sigma(Z)$$

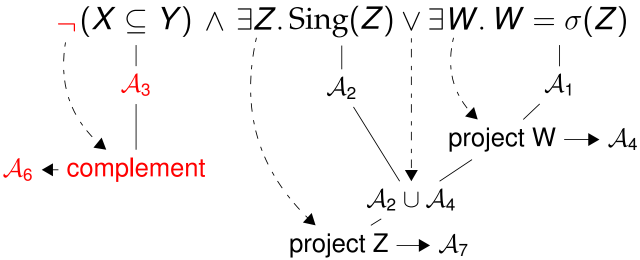


Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$

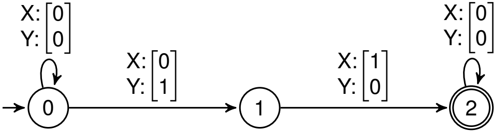


- Example:

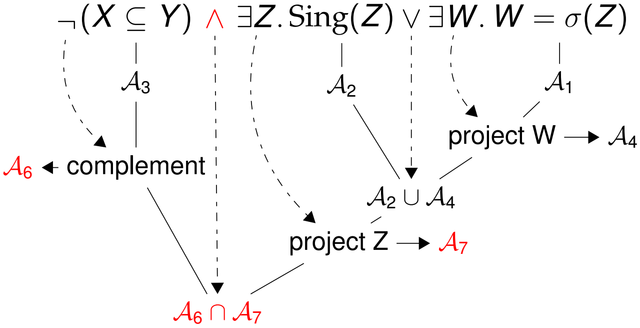


Deciding WS1S using deterministic automata

- example of base automaton for $X = \sigma(Y)$

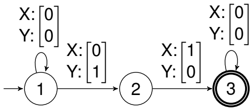


- Example:



How to handle quantification

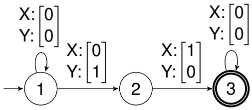
- issue with projection (existential quantification)
  - after removing of the tracks not all models would be accepted
  - so we need to adjust the final states



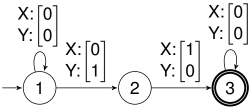
$\mathcal{A}_{X=\sigma(Y)}$

How to handle quantification

- issue with projection (existential quantification)
  - after removing of the tracks not all models would be accepted
  - so we need to adjust the final states



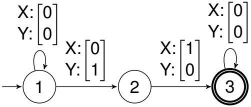
$\mathcal{A}_{X=\sigma(Y)}$



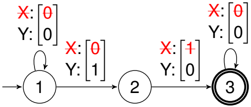
$\rightarrow$ Projection

How to handle quantification

- issue with projection (existential quantification)
  - after removing of the tracks not all models would be accepted
  - so we need to adjust the final states



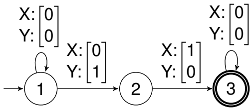
$\mathcal{A}_{X=\sigma(Y)}$



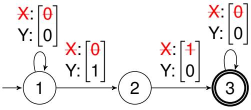
$\rightarrow$ Projection

How to handle quantification

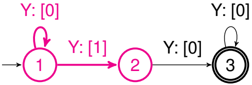
- issue with projection (existential quantification)
  - after removing of the tracks not all models would be accepted
  - so we need to adjust the final states



$\mathcal{A}_{X=\sigma(Y)}$



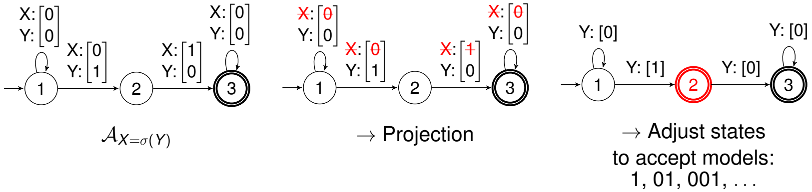
$\rightarrow$ Projection



$\rightarrow$ Adjust states to accept models: 1, 01, 001, ...

How to handle quantification

- issue with projection (existential quantification)
  - after removing of the tracks not all models would be accepted
  - so we need to adjust the final states



Deciding WS1S using non-deterministic automata

- we consider only formulae in Prenex Normal Form (∃PNF)
  - we focus on dealing with prefix and alternations of quantifications
- based on number of alternations m

$$\underbrace{\varphi = \neg \exists \mathcal{X}_m \neg \ldots \neg \exists \mathcal{X}_2 \underbrace{\neg \exists \mathcal{X}_1 : \varphi_0(\mathbb{X})}_{\varphi_1}}_{\varphi_m} \tag{1}$$

Deciding WS1S using non-deterministic automata

- we consider only formulae in Prenex Normal Form (∃PNF)
  - we focus on dealing with prefix and alternations of quantifications
- based on number of alternations m

$$\underbrace{\varphi = \neg\exists\mathcal{X}_m \neg \ldots \neg\exists\mathcal{X}_2 \underbrace{\neg\exists\mathcal{X}_1 : \varphi_0(\mathbb{X})}_{\varphi_1}}_{\varphi_m} \tag{1}$$

$\rightarrow$ hierarchical family of automata defined as follows:

- $\mathcal{A}_{\varphi_0}$ = by composition of atomic automata (previously described)
- $\mathcal{A}_{\varphi_m} = (\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^{Q_0}}}}}}}_{m}, \Delta_m, I_m, F_m)$

The intuition behind the procedure

Key observation for ground formulae

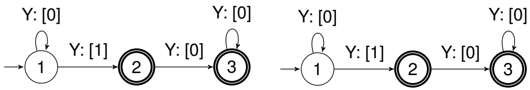
$$\varphi \models \text{ iff } I_m \cap F_m \neq \emptyset$$

The intuition behind the procedure

Key observation for ground formulae

$$\varphi \models \text{ iff } I_m \cap F_m \neq \emptyset$$

- Why?
  - eventually the symbols degenerate to empty ones ...



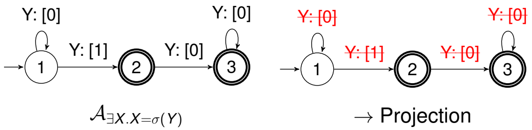
$\mathcal{A}_{\exists X.X=\sigma(Y)}$ $\rightarrow$ Projection

The intuition behind the procedure

Key observation for ground formulae

$$\varphi \models \text{iff } I_m \cap F_m \neq \emptyset$$

- Why?
  - eventually the symbols degenerate to empty ones . . .

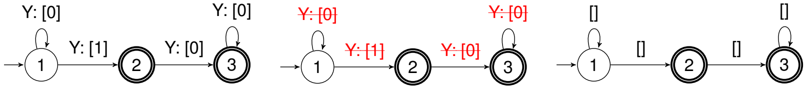


The intuition behind the procedure

Key observation for ground formulae

$$\varphi \models \text{ iff } I_m \cap F_m \neq \emptyset$$

- Why?
  - eventually the symbols degenerate to empty ones . . .



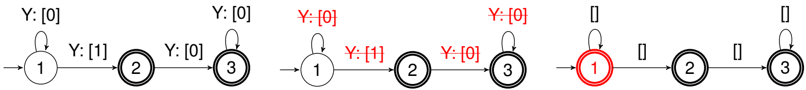
$\mathcal{A}_{\exists X.X=\sigma(Y)}$ $\rightarrow$ Projection $\mathcal{A}_{\exists Y,X.X=\sigma(Y)}$

The intuition behind the procedure

Key observation for ground formulae

$$\varphi \models \text{iff } I_m \cap F_m \neq \emptyset$$

- Why?
  - eventually the symbols degenerate to empty ones ...



$\mathcal{A}_{\exists X.X=\sigma(Y)}$ $\rightarrow$ Projection $\mathcal{A}_{\exists Y,X.X=\sigma(Y)}$

Construction of initial states I_m

- Constructing the whole automaton for φ_m is unnecessary!
  - we construct initial/final states only
  - and test whether they intersect

Construction of initial states I_m

- Constructing the whole automaton for φ_m is unnecessary!
  - we construct initial/final states only
  - and test whether they intersect
- construction of initial states is straightforward; starting from I_0:

Construction of initial states I_m

- Constructing the whole automaton for φ_m is unnecessary!
  - we construct initial/final states only
  - and test whether they intersect
- construction of initial states is straightforward; starting from I_0:
  - $I_1 = \{I_0\}$

Construction of initial states I_m

- Constructing the whole automaton for φ_m is unnecessary!
  - we construct initial/final states only
  - and test whether they intersect
- construction of initial states is straightforward; starting from I_0:
  - $I_1 = \{I_0\}$
  - $I_2 = \{I_1\} = \{\{I_0\}\}$

Construction of initial states I_m

- Constructing the whole automaton for φ_m is unnecessary!
  - we construct initial/final states only
  - and test whether they intersect

- construction of initial states is straightforward; starting from I_0:
  - $I_1 = \{I_0\}$
  - $I_2 = \{I_1\} = \{\{I_0\}\}$

  $\vdots$
  - $I_m = \{I_{m-1}\} = \underbrace{\{\{\ldots\{}_{m} I_0\}\ldots\}\}$
    - based on determinisation procedure

Construction of initial states I_m

- Constructing the whole automaton for φ_m is unnecessary!
  - we construct initial/final states only
  - and test whether they intersect

- construction of initial states is straightforward; starting from I_0:
  - $I_1 = \{I_0\}$
  - $I_2 = \{I_1\} = \{\{I_0\}\}$

    $\vdots$
  - $I_m = \{I_{m-1}\} = \underbrace{\{\{\ldots\{}_{m} I_0\}\ldots\}\}$
    - based on determinisation procedure

- final states are more tricky
  - issue with projection (previously described)
  - multiple levels of determinisation

Introduction to the computation of final states

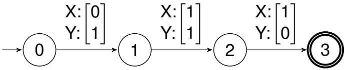
- we already have:
  - formula in ∃PNF: $\varphi = \neg \exists \mathcal{X}_m \neg \ldots \neg \exists \mathcal{X}_2 \neg \exists \mathcal{X}_1 : \varphi_0(\mathbb{X})$
  - base automaton for φ_0

Introduction to the computation of final states

- we already have:
  - formula in ∃PNF: $\varphi = \neg\exists\mathcal{X}_m\neg\ldots\neg\exists\mathcal{X}_2\neg\exists\mathcal{X}_1 : \varphi_0(\mathbb{X})$
  - base automaton for φ_0
- our proposed method
  - is based on generalized backward reachability of final states
  - works on symbolic representation of states, sets of states, sets of sets of states ...
    - for final states $\rightarrow$ compute their predecessors pre_0
      (Intuition) states reaching final states become non-final after negation
    - for non-final states $\rightarrow$ compute their controllable predecessors $cpre_0$
      (Intuition) states leading outside of non-final states become final after negation
  - prunes states on all levels of the hierarchy to achieve minimal representation

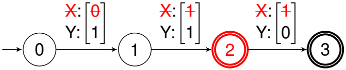
Towards symbolic representation

- Motivating example: $\neg\exists X.\varphi$
  - $Q = \{0, 1, 2, 3\}$
  - $F = \{3\}$



Towards symbolic representation

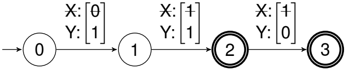
- Motivating example: $\neg\exists X.\varphi$
  - $Q = \{0, 1, 2, 3\}$
  - $F = \{3\}$



- After projection:
  - $F^{\exists} = \{2, 3\}$
  - $N^{\exists} = Q \setminus F^{\exists} = \{0, 1\}$

Towards symbolic representation

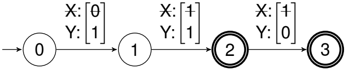
- Motivating example: $\neg\exists X.\varphi$
  - $Q = \{0, 1, 2, 3\}$
  - $F = \{3\}$



- After projection:
  - $F^{\exists} = \{2, 3\}$
  - $N^{\exists} = Q \setminus F^{\exists} = \{0, 1\}$
- After negation:
  - $F_1 = F_{\neg\exists} = \{\{0\}, \{1\}, \{0, 1\}\}$
  - $N_1 = \{\{2\}, \{3\}, \{2, 0\}, \{3, 0\}, \ldots \{2, 3, 0\}, \{2, 3, 1\}, \ldots \{0, 1, 2, 3\}\}$

Towards symbolic representation

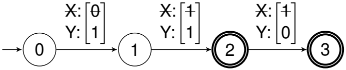
- Motivating example: $\neg\exists X.\varphi$
  - $Q = \{0, 1, 2, 3\}$
  - $F = \{3\}$



- After projection:
  - $F^{\exists} = \{2, 3\}$
  - $N^{\exists} = Q \setminus F^{\exists} = \{0, 1\}$
- After negation:
  - $F_1 = F_{\neg\exists} = \{\{0\}, \{1\}, \{0, 1\}\}$
  - $N_1 = \{\{2\}, \{3\}, \{2, 0\}, \{3, 0\}, \ldots \{2, 3, 0\}, \{2, 3, 1\}, \ldots \{0, 1, 2, 3\}\}$

Towards symbolic representation

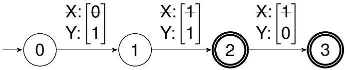
- Motivating example: $\neg\exists X.\varphi$
  - $Q = \{0, 1, 2, 3\}$
  - $F = \{3\}$



- After projection:
  - $F^{\exists} = \{2, 3\}$
  - $N^{\exists} = Q \setminus F^{\exists} = \{0, 1\}$
- After negation:
  - $F_1 = F_{\neg\exists} = \{\{0\}, \{1\}, \{0, 1\}\}$
    $= \downarrow \{\{0, 1\}\}$
  - $N_1 = \{\{2\}, \{3\}, \{2, 0\}, \{3, 0\}, \ldots \{2, 3, 0\}, \{2, 3, 1\}, \ldots \{0, 1, 2, 3\}\}$
    $= \uparrow \{\{2\}, \{3\}\}$

Towards symbolic representation

- Motivating example: $\neg\exists X.\varphi$
  - $Q = \{0, 1, 2, 3\}$
  - $F = \{3\}$



- After projection:
  - $F^{\exists} = \{2, 3\}$
  - $N^{\exists} = Q \setminus F^{\exists} = \{0, 1\}$
- After negation:
  - $F_1 = F_{\neg\exists} = \{\{0\}, \{1\}, \{0, 1\}\}$
    $= \downarrow \{\{0, 1\}\}$
  - $N_1 = \{\{2\}, \{3\}, \{2, 0\}, \{3, 0\}, \ldots \{2, 3, 0\}, \{2, 3, 1\}, \ldots \{0, 1, 2, 3\}\}$
    $= \uparrow \{\{2\}, \{3\}\}$
- so why not work with this symbolic representation only?

Computing final states F_m of formula φ_m

- Given $\varphi = \neg \exists \mathcal{X}_m \neg \ldots \neg \exists \mathcal{X}_2 \neg \exists \mathcal{X}_1 : \varphi_0(\mathbb{X})$

Computing final states F_m of formula φ_m

- Given $\varphi = \neg \exists \mathcal{X}_m \neg \ldots \neg \exists \mathcal{X}_2 \neg \exists \mathcal{X}_1 : \varphi_0(\mathbb{X})$

1. Extend set of final states after $\exists$: $F_0^{\exists} = \{\mu Z. F \cup \mathtt{pre}_0(Z)\}$

Computing final states F_m of formula φ_m

- Given $\varphi = \neg \exists \mathcal{X}_m \neg \ldots \neg \exists \mathcal{X}_2 \neg \exists \mathcal{X}_1 : \varphi_0(\mathbb{X})$

1. Extend set of final states after $\exists$: $F_0^{\exists} = \{\mu Z . F \cup \mathtt{pre}_0(Z)\}$
2. Negate the final states: $N_1 = \uparrow \{F_0^{\exists}\}$

Computing final states F_m of formula φ_m

- Given $\varphi = \neg \exists \mathcal{X}_m \neg \ldots \neg \exists \mathcal{X}_2 \neg \exists \mathcal{X}_1 : \varphi_0(\mathbb{X})$

1. Extend set of final states after $\exists$: $F_0^{\exists} = \{\mu Z. F \cup \mathtt{pre}_0(Z)\}$
2. Negate the final states: $N_1 = \uparrow \{F_0^{\exists}\}$
3. Reduce set of non-final states after $\exists$: $N_1^{\exists} = \{\nu Z. N_1 \cap \mathtt{cpre}_0(Z)\}$
    - Notice the duality with step 1.

$$\cap \;\mapsto\; \cup \qquad \mathtt{cpre}_0 \;\mapsto\; \mathtt{pre}_0 \qquad \nu \;\mapsto\; \mu \tag{2}$$

Computing final states F_m of formula φ_m

- Given $\varphi = \neg\exists\mathcal{X}_m\neg\ldots\neg\exists\mathcal{X}_2\neg\exists\mathcal{X}_1 : \varphi_0(\mathbb{X})$

1. Extend set of final states after $\exists$: $F_0^{\exists} = \{\mu Z.F \cup \mathtt{pre}_0(Z)\}$
2. Negate the final states: $N_1 =\uparrow\{F_0^{\exists}\}$
3. Reduce set of non-final states after $\exists$: $N_1^{\exists} = \{\nu Z.N_1 \cap \mathtt{cpre}_0(Z)\}$
   - Notice the duality with step 1.

$$\cap \mapsto \cup \qquad \mathtt{cpre}_0 \mapsto \mathtt{pre}_0 \qquad \nu \mapsto \mu \tag{2}$$

4. Negate the non-final states: $F_2 =\downarrow\{N_1^{\exists}\}$

Computing final states F_m of formula φ_m

- Given $\varphi = \neg\exists\mathcal{X}_m\neg\ldots\neg\exists\mathcal{X}_2\neg\exists\mathcal{X}_1 : \varphi_0(\mathbb{X})$

1. Extend set of final states after $\exists$: $F_0^{\exists} = \{\mu Z.F \cup \mathtt{pre}_0(Z)\}$
2. Negate the final states: $N_1 =\uparrow \{F_0^{\exists}\}$
3. Reduce set of non-final states after $\exists$: $N_1^{\exists} = \{\nu Z.N_1 \cap \mathtt{cpre}_0(Z)\}$
   - Notice the duality with step 1.

$$\cap \;\mapsto\; \cup \qquad \mathtt{cpre}_0 \;\mapsto\; \mathtt{pre}_0 \qquad \nu \;\mapsto\; \mu \tag{2}$$

4. Negate the non-final states: $F_2 =\downarrow \{N_1^{\exists}\}$

$\vdots$

5. and keep alternating between computing final and non-final states until F_m as follows:
   - $F_{i+1} =\downarrow \{\nu Z.N_i \cap \mathtt{cpre}_0(Z)\}$
   - $N_{i+1} =\uparrow \{\mu Z.F_i \cup \mathtt{pre}_0(Z)\}$

Computing predecessors of the state

- Can we compute $cpre_0/pre_0$ of symbolic states?

Computing predecessors of the state

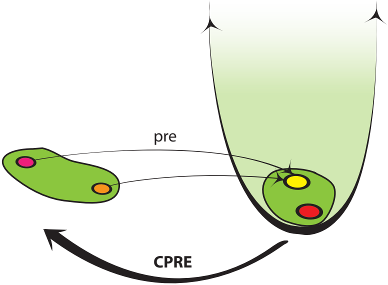
- Can we compute *cpre*$_0$/*pre*$_0$ of symbolic states? Yes!

Computing predecessors of the state

- Can we compute *cpre*$_0$/*pre*$_0$ of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow\{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



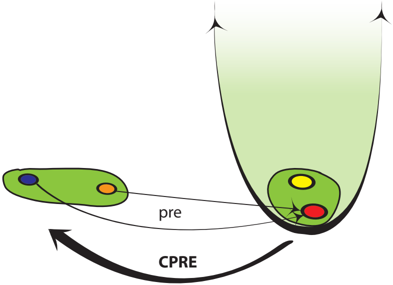
- note that we define the dual lemma for downward closed sets

Computing predecessors of the state

- Can we compute *$cpre_0$*/*pre_0* of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow\{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



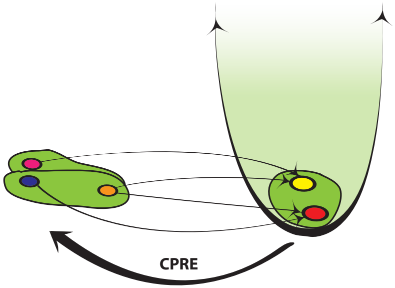
- note that we define the dual lemma for downward closed sets

Computing predecessors of the state

- Can we compute $cpre_0/pre_0$ of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow \{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



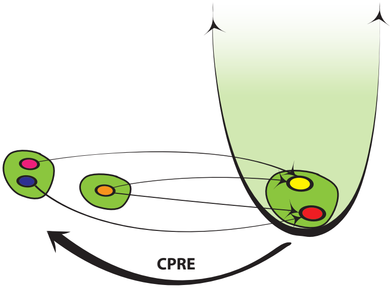
- note that we define the dual lemma for downward closed sets

Computing predecessors of the state

- Can we compute *cpre*$_0$/*pre*$_0$ of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow\{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



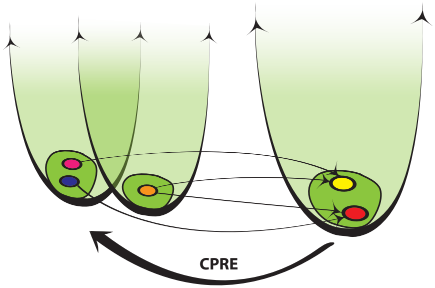
- $\coprod$ breaks the predecessors into new generators that cover the Q
- note that we define the dual lemma for downward closed sets

Computing predecessors of the state

- Can we compute *cpre*$_0$/*pre*$_0$ of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow\{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



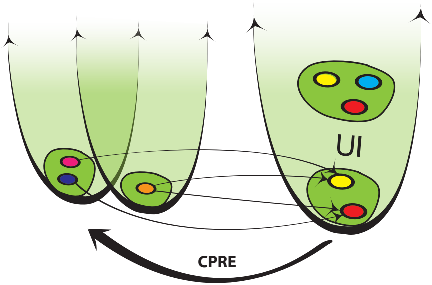
- $\coprod$ breaks the predecessors into new generators that cover the Q
- note that we define the dual lemma for downward closed sets

Computing predecessors of the state

- Can we compute *cpre*$_0$/*pre*$_0$ of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow\{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



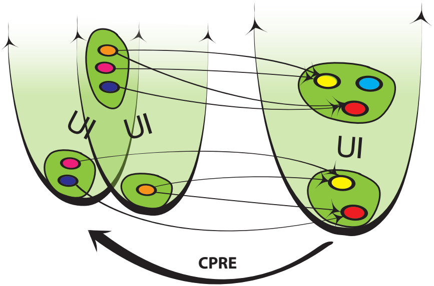
- $\coprod$ breaks the predecessors into new generators that cover the Q
- note that we define the dual lemma for downward closed sets

Computing predecessors of the state

- Can we compute $cpre_0/pre_0$ of symbolic states? Yes!

Lemma. 1

$$\mathtt{cpre}_0(\uparrow\{Q\}) = \uparrow \coprod \{\mathtt{pre}_0(Q)\}$$



- $\coprod$ breaks the predecessors into new generators that cover the Q
- note that we define the dual lemma for downward closed sets

How to achieve state space reduction

- We showed the nested structure of F_m is very complex,

How to achieve state space reduction

- We showed the nested structure of F_m is very complex,
  - but we only work with the symbolic representation of the generators (with antichains)
  - ... and the generators of the generators and ...
  - this itself is the first source of space reduction

How to achieve state space reduction

- We showed the nested structure of F_m is very complex,
  - but we only work with the symbolic representation of the generators (with antichains)
  - ...and the generators of the generators and ...
  - this itself is the first source of space reduction
- further we prune the generators subsumed by other generators
  - the subsumption relation is computed on nested structure of symbolic representation of lower levels

Experimental results

- implemented in dWiNA
- compared with MONA:
  - on generated and real formulae
  - in generic and ∃PNF form

	MONA				dWiNA	
	Time [s]		Space [states]		Time [s]	Space [states]
real	normal	∃PNF	normal	∃PNF	Prefix	Prefix
list-reverse-after-loop	0.01	0.01	179	1 326	0.01	100
list-reverse-in-loop	0.02	0.47	1 311	70 278	0.02	260
bubblesort-else	0.01	0.45	1 285	12 071	0.01	14
bubblesort-if-else	0.02	2.17	4 260	116 760	0.23	234
bubblesort-if-if	0.12	5.29	8 390	233 372	1.14	28
generated						
3 alternations	-	0.57	-	60 924	0.01	50
4 alternations	-	1.79	-	145 765	0.02	58
5 alternations	-	4.98	-	349 314	0.02	70
6 alternations	-	**TO**	-	**TO**	0.47	90

Conclusion and Future Work

- Future work
  - extension to WS2S
    - opens whole new world of tree structures
  - generalization of symbolic tree representation
    - to process logical connectives
    - to handle general (non-∃PNF) formulae

- Conclusion
  - WS1S = Great expressivity, yet decidable!
  - Novel approach based on antichains
  - Encouraging results in terms of space reduction

Thank you for your attention!

Any questions?