Logico-Numerical Abstract Acceleration and Application to the Verification of Data-Flow Programs

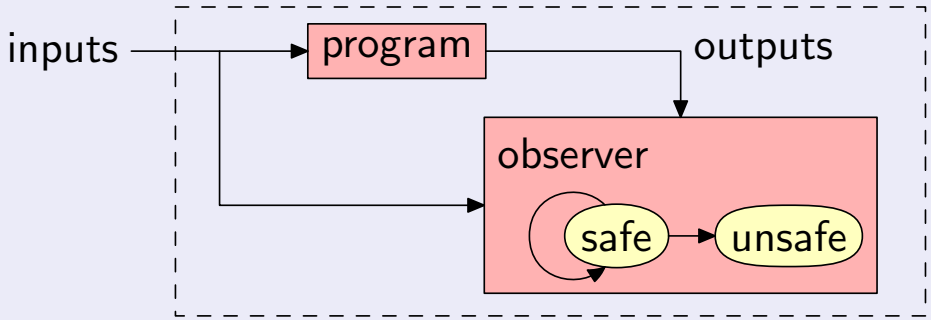
Peter Schrammel and Bertrand Jeannet
{peter.schrammel,bertrand.jeannet}@inria.fr

INRIA Grenoble – Rhône-Alpes

The 18th International Static Analysis Symposium
September 14-16, 2011, Venice, Italy

Motivation

Verification of safety properties about LUSTRE programs



- Check by reachability analysis whether "safe" is an invariant

Observations and goals

- Large Boolean state space
  - $\longrightarrow$ Avoid enumeration: use state space partitioning
- Mostly regular linear arithmetic
  - $\longrightarrow$ Exploit a specialized method: abstract acceleration

Reachability analysis of numerical programs

Abstract interpretation (Cousot & Cousot 1977)	over-approximation (e.g. intervals, octagons, polyhedra)	termination
Acceleration (Finkel & Leroux 2002)	exact (Presburger formula)	no guarantee for termination for non-flat systems

Reachability analysis of numerical programs

Abstract interpretation (Cousot & Cousot 1977)	over-approximation (e.g. intervals, octagons, polyhedra)	termination
Acceleration (Finkel & Leroux 2002)	exact (Presburger formula)	no guarantee for termination for non-flat systems

What is acceleration?

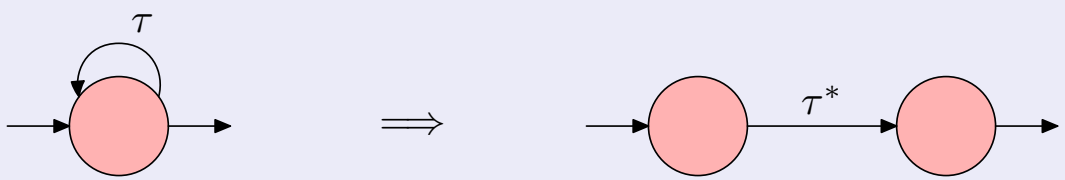
- Method for analyzing loops in control flow graphs (CFG) of numerical programs
- Replace a loop transition τ by its reflexive and transitive closure τ^*

Reachability analysis of numerical programs

Abstract interpretation (Cousot & Cousot 1977)	over-approximation (e.g. intervals, octagons, polyhedra)	termination
Acceleration (Finkel & Leroux 2002)	exact (Presburger formula)	no guarantee for termination for non-flat systems

What is acceleration?

- Method for analyzing loops in control flow graphs (CFG) of numerical programs
- Replace a loop transition τ by its reflexive and transitive closure τ^*



- Applicable to affine transitions $\tau : \mathbf{Ax} \leq \mathbf{b} \rightarrow \mathbf{x}' = \mathbf{Cx} + \mathbf{d}$
  - where $\{\mathbf{C}^p \mid p \geq 0\}$ is finite

Combination of acceleration and abstract interpretation

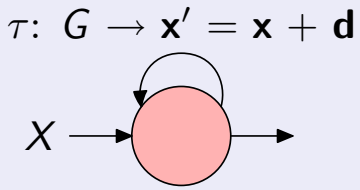
Abstract Acceleration (Gonnord & Halbwachs 2006)

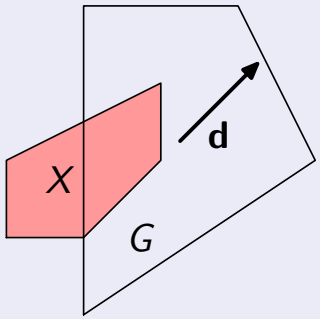
- Computes a convex polyhedron $\tau^{\otimes}$ over-approximating the exact result τ^*

Combination of acceleration and abstract interpretation

Abstract Acceleration (Gonnord & Halbwachs 2006)

- Computes a convex polyhedron $\tau^{\otimes}$ over-approximating the exact result τ^*

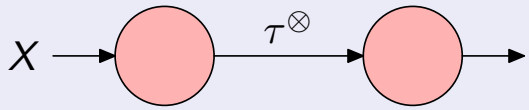


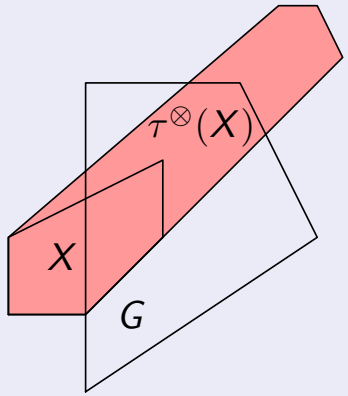


Combination of acceleration and abstract interpretation

Abstract Acceleration (Gonnord & Halbwachs 2006)

- Computes a convex polyhedron $\tau^{\otimes}$ over-approximating the exact result τ^*

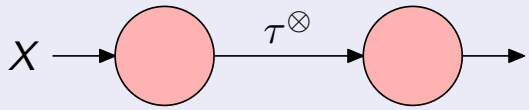


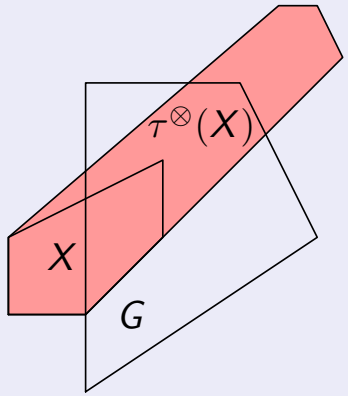


Combination of acceleration and abstract interpretation

Abstract Acceleration (Gonnord & Halbwachs 2006)

- Computes a convex polyhedron $\tau^{\otimes}$ over-approximating the exact result τ^*





- Improves precision:
  - Computes $\tau^{\otimes} \approx (\alpha \circ \tau^* \circ \gamma)$ instead of $(\alpha \circ \tau \circ \gamma)^*$
  - No widening for flat systems
- Improves performance: fewer iterations

Application to LUSTRE programs?

Issues

- Numerical input variables $\surd$ (Schrammel & Jeannet 2010)
- Boolean state variables

Approaches

	abstract interpretation	
enumeration of Boolean states	*classical AI*	

Application to LUSTRE programs?

Issues

- Numerical input variables $\surd$ (Schrammel & Jeannet 2010)
- Boolean state variables

Approaches

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	(Gonnord et al '06)

Application to LUSTRE programs?

Issues

- Numerical input variables $\surd$ (Schrammel & Jeannet 2010)
- Boolean state variables

Approaches

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	(Gonnord et al '06)
state space partitioning	(Jeannet et al '99)	

Application to LUSTRE programs?

Issues

- Numerical input variables $\surd$ (Schrammel & Jeannet 2010)
- Boolean state variables

Approaches

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	(Gonnord et al '06)
state space partitioning	(Jeannet et al '99)	**(this talk)**

Application to LUSTRE programs?

Issues

- Numerical input variables $\surd$ (Schrammel & Jeannet 2010)
- Boolean state variables

Approaches

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	(Gonnord et al '06)
state space partitioning	(Jeannet et al '99)	**(this talk)**

Our approach

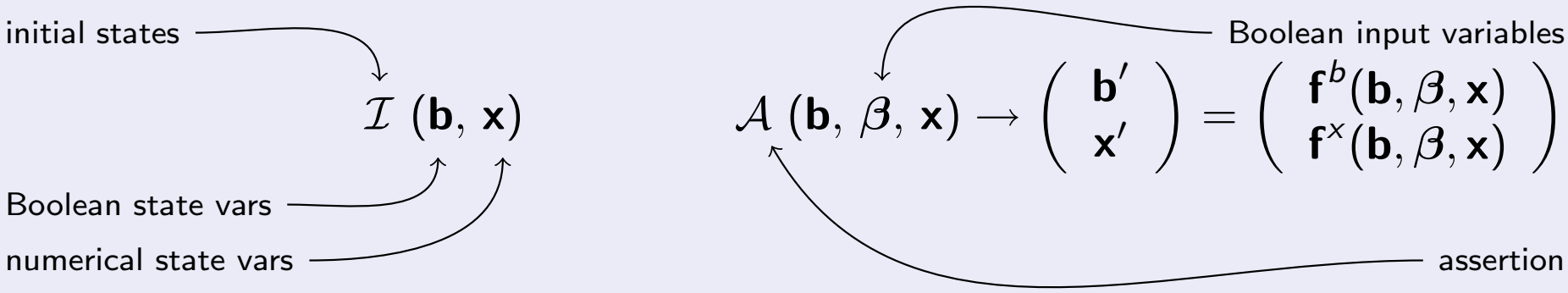
Using a logico-numerical abstract domain:

- Logico-numerical abstract acceleration method
- Partitioning heuristics

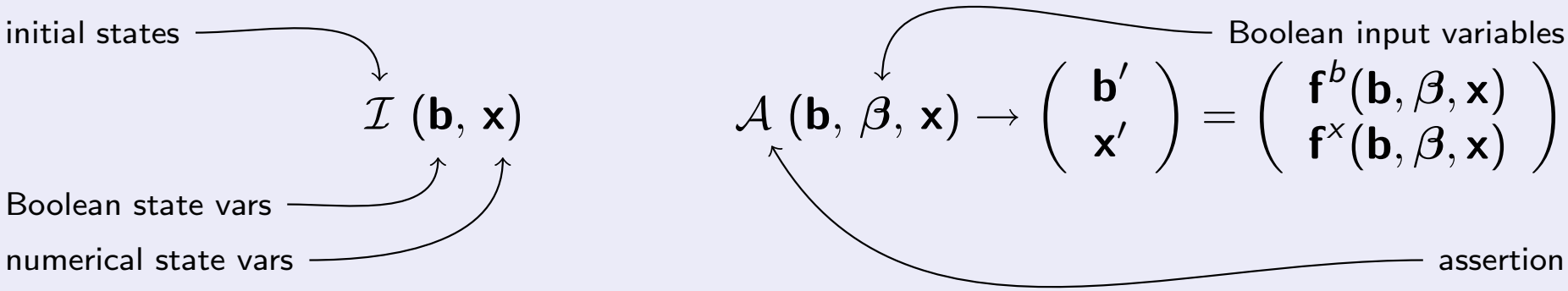
Outline

Outline

Program Model



Program Model



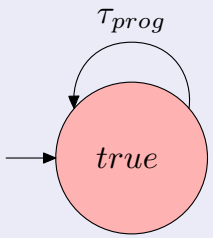
Example

$\mathcal{I}(\mathbf{b}, \mathbf{x}) = \neg b_0 \wedge \neg b_1 \wedge x_0 = 0 \wedge x_1 = 0 \wedge x_2 = 0$

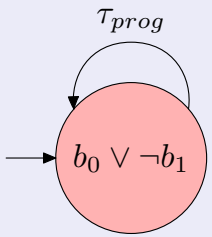
$true \rightarrow$

$$\begin{cases} b_0' = b_0 \vee (\neg b_0 \wedge x_0 > 10 \wedge x_1 > 10) \\ b_1' = b_1 \vee (\neg b_1 \wedge x_0 > 20) \\ x_0' = \begin{cases} x_0 + 1 & \text{if } (\neg b_0 \wedge \neg b_1 \wedge x_0 \leq 10 \wedge \beta) \vee (b_0 \wedge \neg b_1 \wedge x_0 \leq 20) \\ 0 & \text{if } \neg b_0 \wedge \neg b_1 \wedge x_0 > 10 \wedge x_1 > 10 \\ x_0 & \text{otherwise} \end{cases} \\ x_1' = \begin{cases} x_1 + 1 & \text{if } \neg b_0 \wedge \neg b_1 \wedge x_1 \leq 10 \wedge \neg\beta \\ x_1 & \text{otherwise} \end{cases} \\ x_2' = \begin{cases} x_2 + 1 & \text{if } (\neg b_0 \wedge \neg b_1 \wedge (x_0 \leq 10 \wedge \beta \vee x_1 \leq 10 \wedge \neg\beta)) \vee (b_0 \wedge \neg b_1) \\ x_2 & \text{otherwise} \end{cases} \end{cases}$$

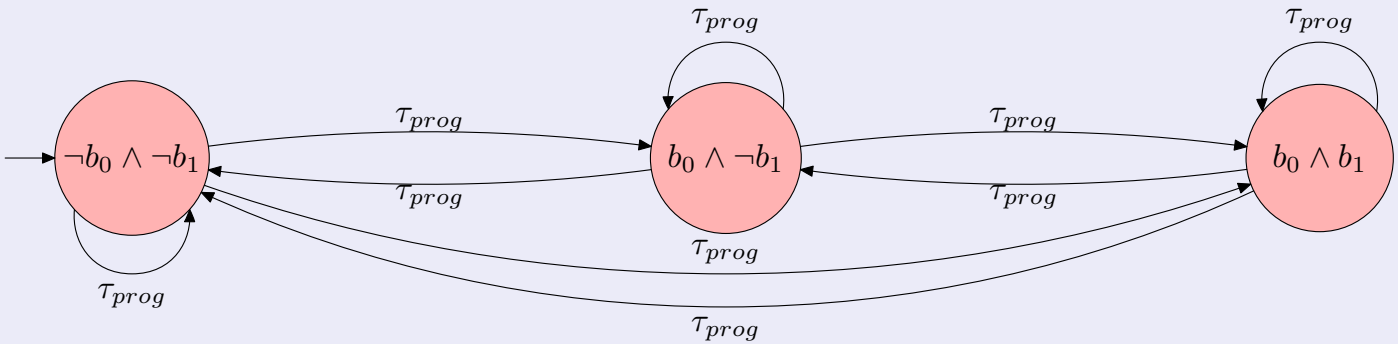
Classical Approach



Classical Approach

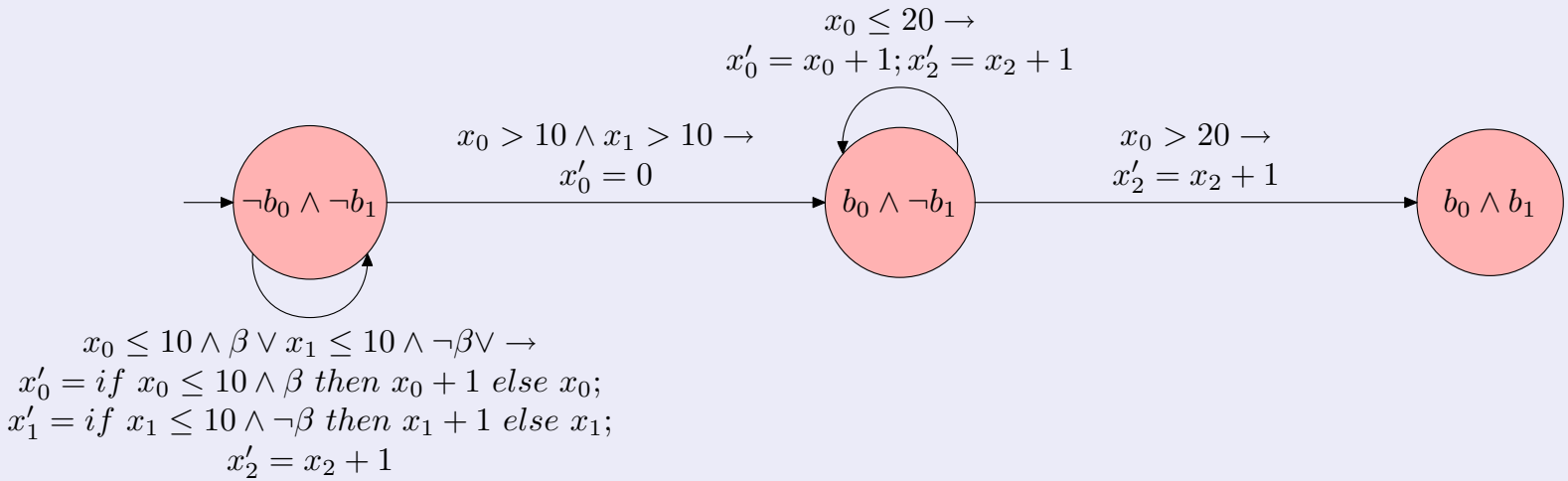


Classical Approach



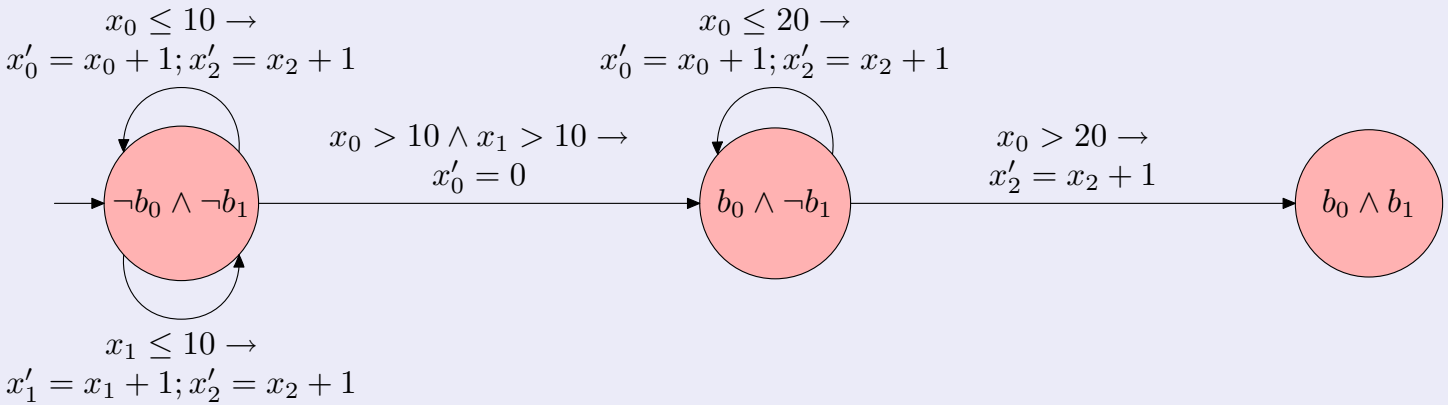
1. Boolean state space enumeration

Classical Approach



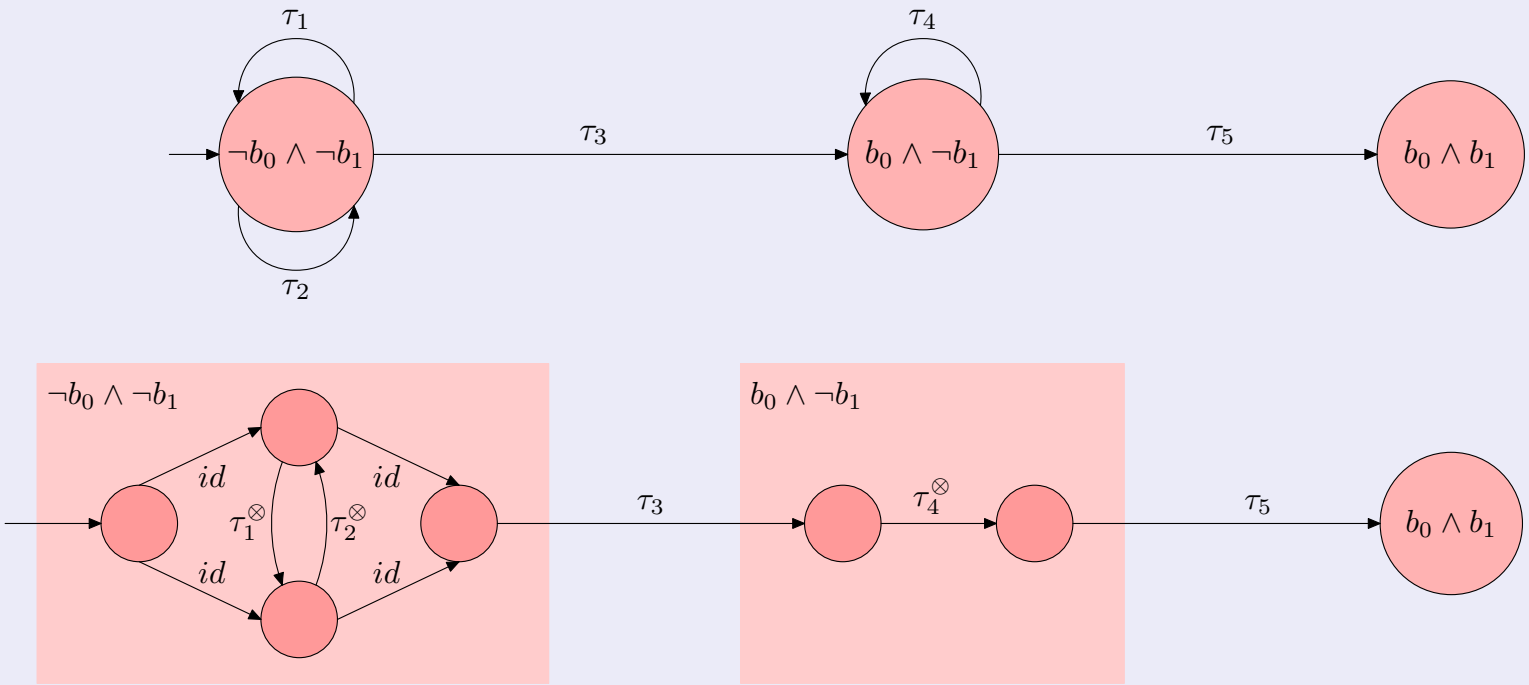
1. Boolean state space enumeration
2. Transition simplification by partial evaluation

Classical Approach



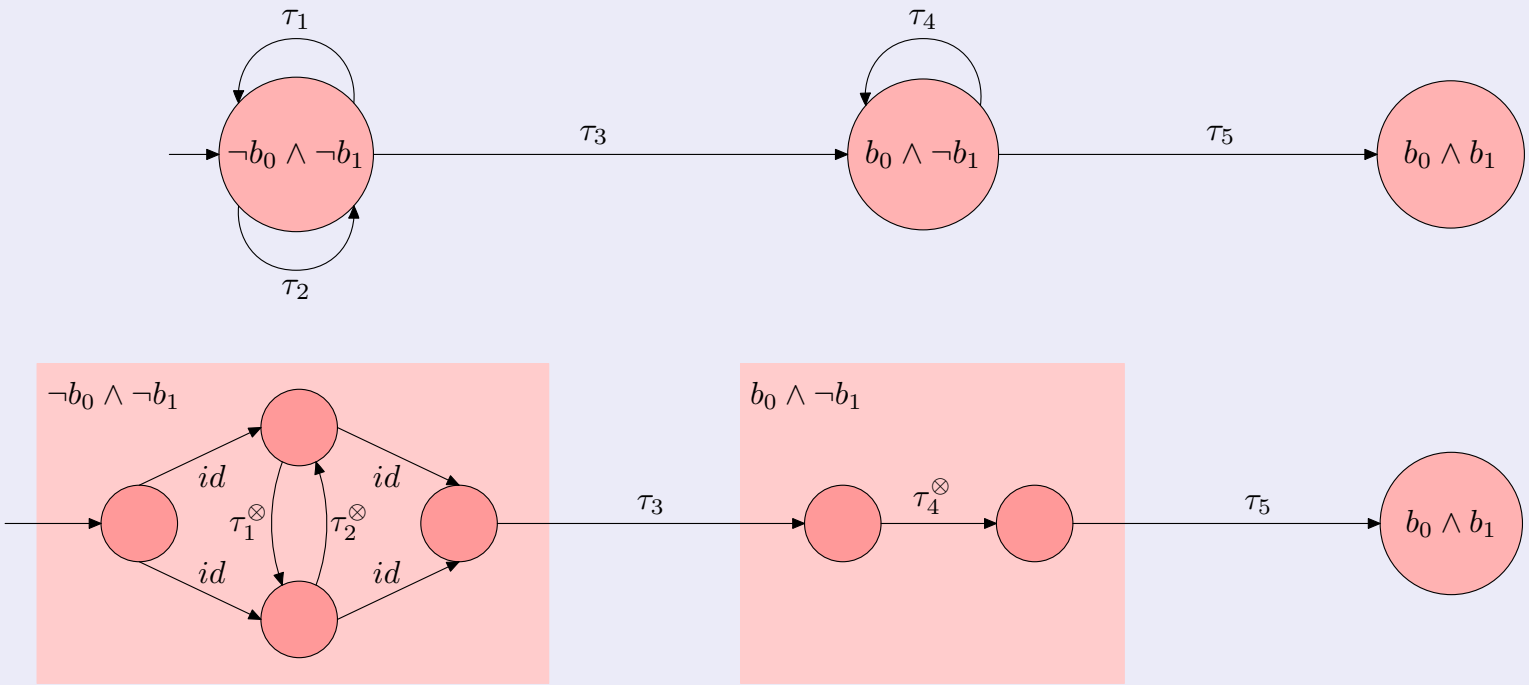
1. Boolean state space enumeration
2. Transition simplification by partial evaluation
3. Elimination of Boolean input variables

Classical Approach



1. Boolean state space enumeration
2. Transition simplification by partial evaluation
3. Elimination of Boolean input variables
4. Expansion ("Flattening") of accelerable self-loops

Classical Approach



1. Boolean state space enumeration
2. Transition simplification by partial evaluation
3. Elimination of Boolean input variables
4. Expansion ("Flattening") of accelerable self-loops
5. Analysis

~~Classical~~ Our Approach

1. Boolean state space ~~enumeration~~ partitioning
2. Transition simplification by partial evaluation
3. Elimination of Boolean input variables
4. Expansion ("Flattening") of logico-numerical accelerable self-loops
5. Analysis

Outline

How to accelerate logico-numerical loops?

Numerically accelerable self-loop

$$\tau : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a}(\mathbf{x}) \end{pmatrix}$$

Boolean part of transition function is identity

accelerable affine expression

How to accelerate logico-numerical loops?

Numerically accelerable self-loop

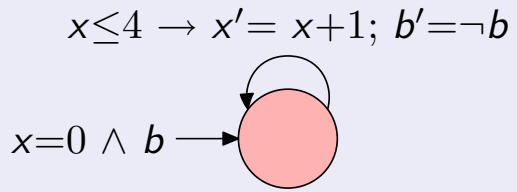
$$\tau : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a}(\mathbf{x}) \end{pmatrix}$$

Boolean part of transition function is identity

accelerable affine expression

Problem:

- Accelerable self-loops might not exist at all
  - $\rightarrow$ Boolean identity too restrictive
  - $\rightarrow$ rarely applicable



How to accelerate logico-numerical loops?

Logico-numerical accelerable self-loop

$$\tau : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \mathbf{f}^b(\mathbf{b}, \mathbf{x}) \\ \mathbf{a}(\mathbf{x}) \end{pmatrix}$$

general Boolean expression

convex

accelerable affine expression

How to accelerate logico-numerical loops?

Logico-numerical accelerable self-loop

$$\tau : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \mathbf{f}^b(\mathbf{b}, \mathbf{x}) \\ \mathbf{a}(\mathbf{x}) \end{pmatrix}$$

general Boolean expression

convex

accelerable affine expression

Idea

Decoupling of numerical and Boolean parts of τ:

$$\tau_x : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a}(\mathbf{x}) \end{pmatrix}$$

$$\tau_b : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} f^b(\mathbf{b}, \mathbf{x}) \\ \mathbf{x} \end{pmatrix}$$

Boolean identity

numerical identity

Logico-Numerical Abstract Acceleration

Propositions

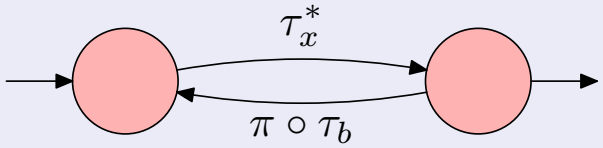
Decoupling abstraction: $\wp(\mathbb{B}^m \times \mathbb{R}^n) \underset{\pi}{\overset{id}{\leftrightarrows}} \wp(\mathbb{B}^m) \times \wp(\mathbb{R}^n)$

Logico-Numerical Abstract Acceleration

Propositions

Decoupling abstraction: $\wp(\mathbb{B}^m \times \mathbb{R}^n) \underset{\pi}{\overset{id}{\leftrightarrows}} \wp(\mathbb{B}^m) \times \wp(\mathbb{R}^n)$

- $\tau^* \subseteq (\pi \circ \tau_b \circ \tau_x^*)^*$

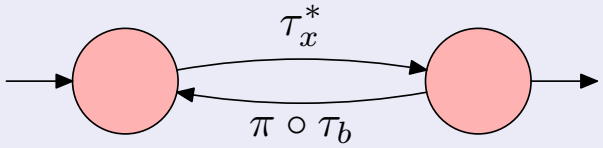


Logico-Numerical Abstract Acceleration

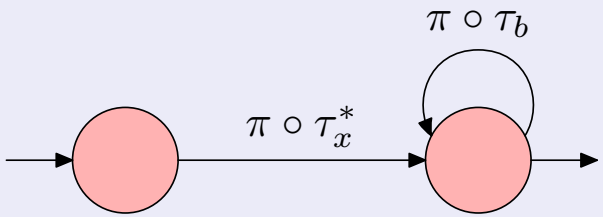
Propositions

Decoupling abstraction: $\wp(\mathbb{B}^m \times \mathbb{R}^n) \underset{\pi}{\overset{id}{\leftrightarrows}} \wp(\mathbb{B}^m) \times \wp(\mathbb{R}^n)$

- $\tau^* \subseteq (\pi \circ \tau_b \circ \tau_x^*)^*$



- $\tau^* \subseteq (\pi \circ \tau_b)^* \circ \pi \circ \tau_x^*$

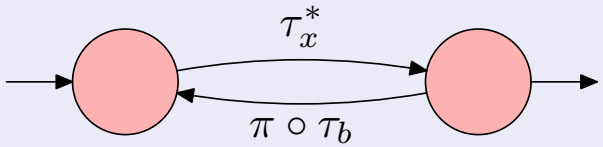


Logico-Numerical Abstract Acceleration

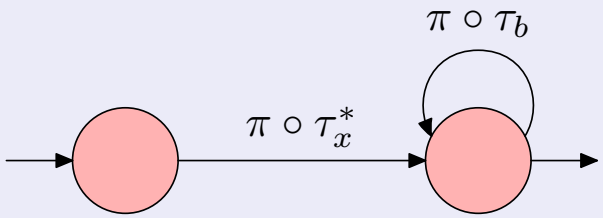
Propositions

Decoupling abstraction: $\wp(\mathbb{B}^m \times \mathbb{R}^n) \underset{\pi}{\overset{id}{\leftrightarrows}} \wp(\mathbb{B}^m) \times \wp(\mathbb{R}^n)$

- $\tau^* \subseteq (\pi \circ \tau_b \circ \tau_x^*)^*$



- $\tau^* \subseteq (\pi \circ \tau_b)^* \circ \pi \circ \tau_x^*$



Logico-numerical accelerable self-loop

general Boolean expression

$$\tau : g^b(\mathbf{b}) \wedge g^x(\mathbf{x}) \rightarrow \begin{pmatrix} \mathbf{b}' \\ \mathbf{x}' \end{pmatrix} = \begin{pmatrix} \mathbf{f}^b(\mathbf{b}, \mathbf{x}) \\ \mathbf{a}(\mathbf{x}) \end{pmatrix}$$

convex

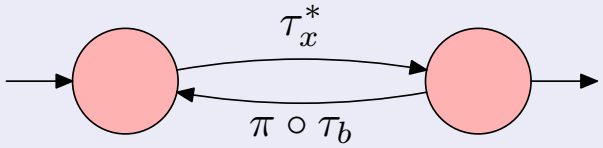
accelerable affine expression

Logico-Numerical Abstract Acceleration

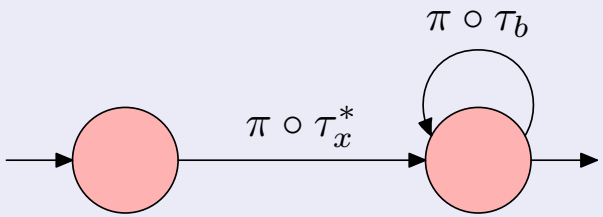
Propositions

Decoupling abstraction: $\wp(\mathbb{B}^m \times \mathbb{R}^n) \underset{\pi}{\overset{id}{\leftrightarrows}} \wp(\mathbb{B}^m) \times \wp(\mathbb{R}^n)$

- $\tau^* \subseteq (\pi \circ \tau_b \circ \tau_x^*)^*$



- $\tau^* \subseteq (\pi \circ \tau_b)^* \circ \pi \circ \tau_x^*$



Theorem

With the abstraction (Jeannet 2003):

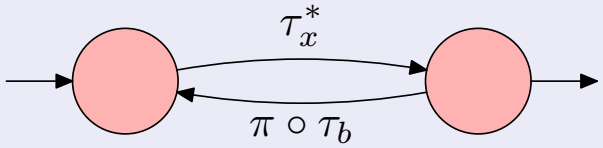
$$\begin{array}{rcl} \wp(\mathbb{B}^m \times \mathbb{R}^n) & \underset{\alpha}{\overset{id}{\leftrightarrows}} & \wp(\mathbb{B}^m) \times Pol(\mathbb{R}^n) \\ S(\mathbf{b}, \mathbf{x}) & \rightarrow & \varphi(\mathbf{b}) \wedge Pol(\mathbf{x}) \end{array}$$

Logico-Numerical Abstract Acceleration

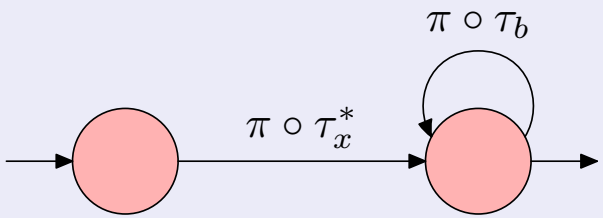
Propositions

Decoupling abstraction: $\wp(\mathbb{B}^m \times \mathbb{R}^n) \underset{\pi}{\overset{id}{\leftrightarrows}} \wp(\mathbb{B}^m) \times \wp(\mathbb{R}^n)$

- $\tau^* \subseteq (\pi \circ \tau_b \circ \tau_x^*)^*$



- $\tau^* \subseteq (\pi \circ \tau_b)^* \circ \pi \circ \tau_x^*$

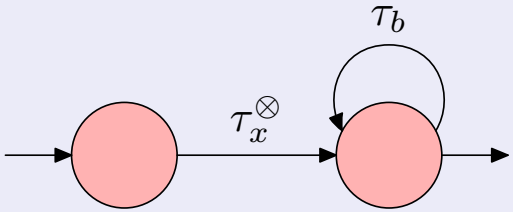


Theorem

With the abstraction (Jeannet 2003):

$$\begin{array}{ccc} \wp(\mathbb{B}^m \times \mathbb{R}^n) & \underset{\alpha}{\overset{id}{\leftrightarrows}} & \wp(\mathbb{B}^m) \times Pol(\mathbb{R}^n) \\ S(\mathbf{b},\mathbf{x}) & \rightarrow & \varphi(\mathbf{b}) \wedge Pol(\mathbf{x}) \end{array}$$

$\tau^* \subseteq \tau^{\otimes}$



Discussion

Advantages

- Accelerate self-loops while fighting state-space explosion using partitioning

Drawback

- Decoupling may lose precision

Discussion

Advantages

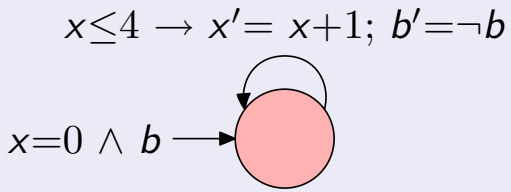
- Accelerate self-loops while fighting state-space explosion using partitioning

Drawback

- Decoupling may lose precision

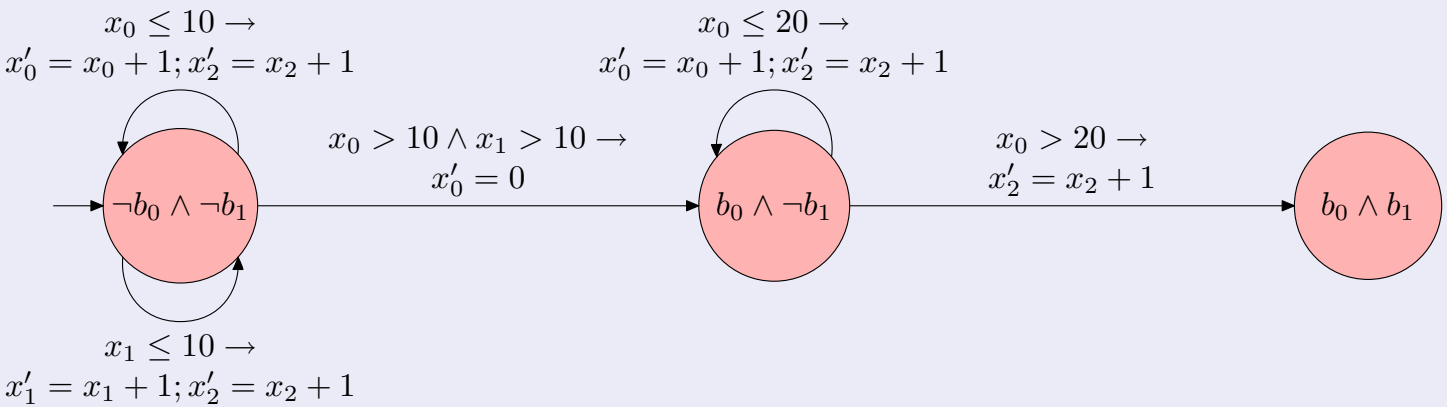
Observations

- Partial decoupling only
- Loses mostly that information that is not representable in the abstract domain anyway
  - Exact reachable set: $\{\mathit{true}\} \times \{0, 2, 4\} \cup \{\mathit{false}\} \times \{1, 3, 5\}$
  - In the abstract domain: $\{\top\} \times \{0 \leq x \leq 5\}$



Discussion

Example: Precision

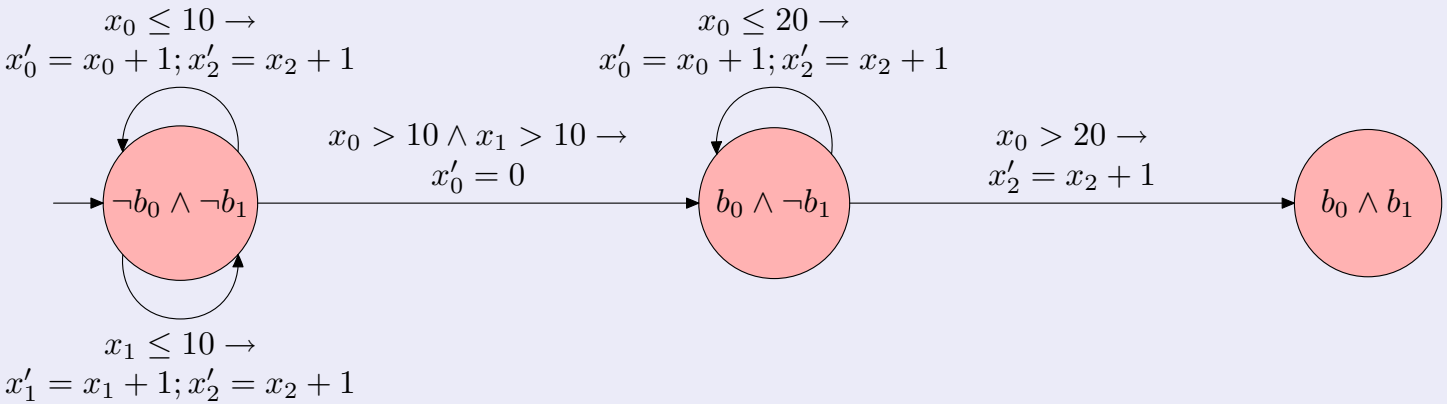


Standard analysis on enumerated CFG

$$0 \leq x_0 \quad \wedge \quad 0 \leq x_1 \quad \wedge \quad x_0 + x_1 \leq x_2$$

Discussion

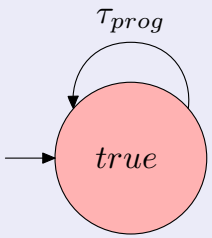
Example: Precision



Standard analysis on enumerated CFG	$0 \leq x_0$	$\wedge$	$0 \leq x_1$	$\wedge$	$x_0 + x_1 \leq x_2$
Abstract acceleration on enumerated CFG	$0 \leq x_0 \leq 21$	$\wedge$	$0 \leq x_1 \leq 11$	$\wedge$	$x_0 + x_1 \leq x_2 \leq 44$

Discussion

Example: Precision



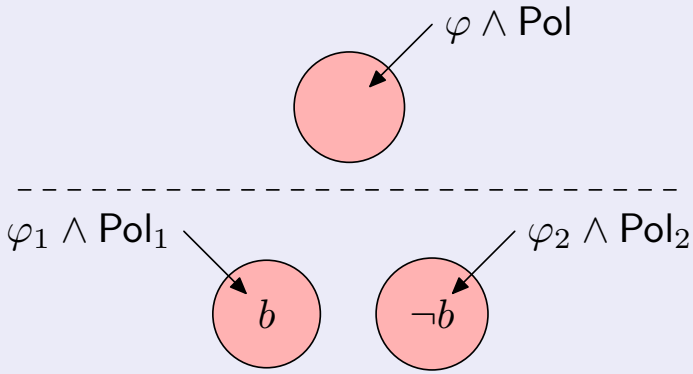
Standard analysis on enumerated CFG	$0 \leq x_0$	$\wedge$	$0 \leq x_1$	$\wedge$	$x_0 + x_1 \leq x_2$
Abstract acceleration on enumerated CFG	$0 \leq x_0 \leq 21$	$\wedge$	$0 \leq x_1 \leq 11$	$\wedge$	$x_0 + x_1 \leq x_2 \leq 44$
Logico-numerical abstract accel. on single location	$0 \leq x_0 \leq 21$	$\wedge$	$0 \leq x_1 \leq 11$	$\wedge$	$x_0 + x_1 \leq x_2$

Outline

Partitioning

Why partitioning?

- Trade-off between performance and precision:
  - Reasonably small CFG to escape state-space explosion
  - Sufficiently large CFG to benefit from:
    - ⋆ $\longrightarrow$ Disjunctive abstract domain
    - ⋆ $\longrightarrow$ Less precision loss in convex unions
    - ⋆ $\longrightarrow$ More targeted application of *widening*



- Limit approximations made by our logico-numerical acceleration method

Partitioning

Our heuristics

- Boolean states with same numerical behavior grouped in same location:
  - $\longrightarrow$ Numerical equations independent of Boolean equations
  - $\longrightarrow$ Reduced impact of decoupling

$$\mathbf{b}_1 \sim \mathbf{b}_2 \Leftrightarrow \begin{cases} \forall \boldsymbol{\beta}_1, \mathcal{C} : & \mathcal{A}(\mathbf{b}_1, \boldsymbol{\beta}_1, \mathcal{C}) \Rightarrow \\ & \exists \boldsymbol{\beta}_2 : \mathcal{A}(\mathbf{b}_2, \boldsymbol{\beta}_2, \mathcal{C}) \wedge f^x(\mathbf{b}_1, \boldsymbol{\beta}_1, \mathcal{C}) = f^x(\mathbf{b}_2, \boldsymbol{\beta}_2, \mathcal{C}) \\ \textit{and} & \textit{vice versa} \end{cases}$$

Partitioning

Our heuristics

- Boolean states with same numerical behavior grouped in same location:
  - $\longrightarrow$ Numerical equations independent of Boolean equations
  - $\longrightarrow$ Reduced impact of decoupling

$$\mathbf{b}_1 \sim \mathbf{b}_2 \Leftrightarrow \left\{ \begin{array}{ll} \forall \boldsymbol{\beta}_1, \mathcal{C} : & \mathcal{A}(\mathbf{b}_1, \boldsymbol{\beta}_1, \mathcal{C}) \Rightarrow \\ & \exists \boldsymbol{\beta}_2 : \mathcal{A}(\mathbf{b}_2, \boldsymbol{\beta}_2, \mathcal{C}) \wedge f^x(\mathbf{b}_1, \boldsymbol{\beta}_1, \mathcal{C}) = f^x(\mathbf{b}_2, \boldsymbol{\beta}_2, \mathcal{C}) \\ \textit{and} & \textit{vice versa} \end{array} \right.$$

Example

$$x' = \left\{ \begin{array}{ll} x+1 & \text{if } b_1 \wedge \beta_1 \wedge x \leq 10 \\ x+1 & \text{if } b_2 \wedge \beta_2 \wedge x \leq 10 \\ x-2 & \text{if } \neg b_1 \wedge \neg b_2 \wedge x \geq 0 \\ x & \text{otherwise} \end{array} \right.$$

Partitioning

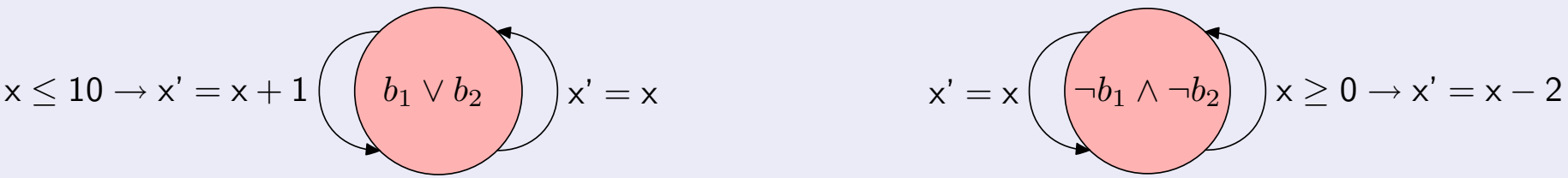
Our heuristics

- Boolean states with same numerical behavior grouped in same location:
  - $\rightarrow$ Numerical equations independent of Boolean equations
  - $\rightarrow$ Reduced impact of decoupling

$$\mathbf{b}_1 \sim \mathbf{b}_2 \Leftrightarrow \begin{cases} \forall \beta_1, \mathcal{C} : & \mathcal{A}(\mathbf{b}_1, \beta_1, \mathcal{C}) \Rightarrow \\ & \exists \beta_2 : \mathcal{A}(\mathbf{b}_2, \beta_2, \mathcal{C}) \wedge f^x(\mathbf{b}_1, \beta_1, \mathcal{C}) = f^x(\mathbf{b}_2, \beta_2, \mathcal{C}) \\ \textit{and} & \textit{vice versa} \end{cases}$$

Example

$$x' = \begin{cases} x+1 & \text{if } b_1 \wedge \beta_1 \wedge x \leq 10 \\ x+1 & \text{if } b_2 \wedge \beta_2 \wedge x \leq 10 \\ x-2 & \text{if } \neg b_1 \wedge \neg b_2 \wedge x \geq 0 \\ x & \text{otherwise} \end{cases}$$



Outline

Experimental Evaluation

Tool NBACCEL

- Uses the abstract domain library BDDAPRON

Comparison

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	ASPIC (Gonnord 2009)
state space partitioning	NBAC (Jeannet 2003) – property-guided incremental partitioning of Boolean and numerical states	NBACCEL (this talk)

Experimental Evaluation

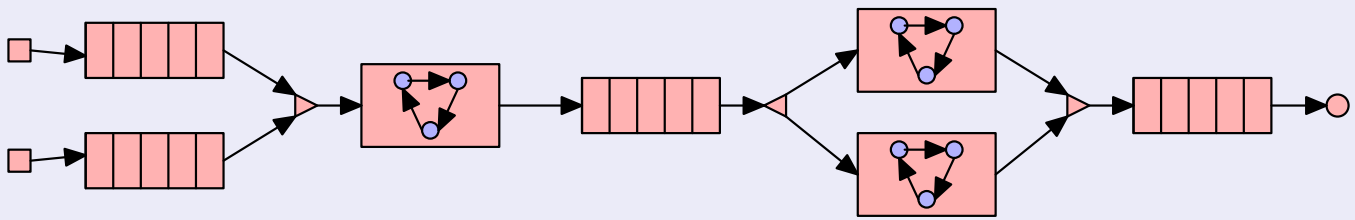
Tool NBACCEL

- Uses the abstract domain library BDDAPRON

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	ASPIC (Gonnord 2009)
state space partitioning	NBAC (Jeannet 2003)	NBACCEL (this talk)

Benchmarks

- Models of production lines: sources, sinks, buffers, machines, etc
- Communication via handshakes: large Boolean state space
- All numerical actions are accelerable
- Properties that depend on numerical variables: e.g. throughput time



Experimental Evaluation

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/**x**/β	size	time	size	time	size	time
Gate 1	4/4/2	7	?	5	**0.73**	24	?
Escalator 1	5/4/2	12	**0.14 (0.04)**	9	0.49	22	?
Traffic 1	4/6/0	18	**0.14 (0.01)**	16	0.19	5	3.49
Traffic 2	4/8/0	18	?	16	**0.35**	28	?
LCM Quest 0a-1	7/2/0	7	**0.04 (0.01)**	5	**0.04**	5	0.05
LCM Quest 0a-2	7/3/0	6	**0.05 (0.01)**	4	**0.05**	8	0.19
LCM Quest 0b-1	10/3/0	19	**0.08 (0.01)**	12	**0.08**	9	?
LCM Quest 0b-2	10/4/0	17	**0.09 (0.01)**	11	0.20	33	?
LCM Quest 0c-1	15/4/0	28	0.17 (0.01)	16	**0.16**	8	0.86
LCM Quest 0c-2	15/5/0	25	**0.20 (0.05)**	14	0.24	50	14.8
LCM Quest 1b-1	16/5/0	55	0.92 (0.04)	29	**0.37**	15	?
LCM Quest 1b-2	16/5/0	45	0.76 (0.12)	23	**0.47**	61	?
LCM Quest 1-1	16/5/0	114	1.99 (0.48)	42	**0.92**	6	2.45
LCM Quest 1-2	16/6/0	100	?	34	?	>156	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation

locations in enumerated CFG

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/**x**/β	size	time	size	time	size	time
Gate 1	4/4/2	7	?	5	**0.73**	24	?
Escalator 1	5/4/2	12	**0.14 (0.04)**	9	0.49	22	?
Traffic 1	4/6/0	18	**0.14 (0.01)**	16	0.19	5	3.49
Traffic 2	4/8/0	18	?	16	**0.35**	28	?
LCM Quest 0a-1	7/2/0	7	**0.04 (0.01)**	5	**0.04**	5	0.05
LCM Quest 0a-2	7/3/0	6	**0.05 (0.01)**	4	**0.05**	8	0.19
LCM Quest 0b-1	10/3/0	19	**0.08 (0.01)**	12	**0.08**	9	?
LCM Quest 0b-2	10/4/0	17	**0.09 (0.01)**	11	0.20	33	?
LCM Quest 0c-1	15/4/0	28	0.17 (0.01)	16	**0.16**	8	0.86
LCM Quest 0c-2	15/5/0	25	**0.20 (0.05)**	14	0.24	50	14.8
LCM Quest 1b-1	16/5/0	55	0.92 (0.04)	29	**0.37**	15	?
LCM Quest 1b-2	16/5/0	45	0.76 (0.12)	23	**0.47**	61	?
LCM Quest 1-1	16/5/0	114	1.99 (0.48)	42	**0.92**	6	2.45
LCM Quest 1-2	16/6/0	100	?	34	?	>156	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation

locations in enumerated CFG

total

analysis only

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/x/β	size	time	size	time	size	time
Gate 1	4/4/2	7	?	5	**0.73**	24	?
Escalator 1	5/4/2	12	**0.14 (0.04)**	9	0.49	22	?
Traffic 1	4/6/0	18	**0.14 (0.01)**	16	0.19	5	3.49
Traffic 2	4/8/0	18	?	16	**0.35**	28	?
LCM Quest 0a-1	7/2/0	7	**0.04 (0.01)**	5	**0.04**	5	0.05
LCM Quest 0a-2	7/3/0	6	**0.05 (0.01)**	4	**0.05**	8	0.19
LCM Quest 0b-1	10/3/0	19	**0.08 (0.01)**	12	**0.08**	9	?
LCM Quest 0b-2	10/4/0	17	**0.09 (0.01)**	11	0.20	33	?
LCM Quest 0c-1	15/4/0	28	0.17 (0.01)	16	**0.16**	8	0.86
LCM Quest 0c-2	15/5/0	25	**0.20 (0.05)**	14	0.24	50	14.8
LCM Quest 1b-1	16/5/0	55	0.92 (0.04)	29	**0.37**	15	?
LCM Quest 1b-2	16/5/0	45	0.76 (0.12)	23	**0.47**	61	?
LCM Quest 1-1	16/5/0	114	1.99 (0.48)	42	**0.92**	6	2.45
LCM Quest 1-2	16/6/0	100	?	34	?	>156	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation

locations in enumerated CFG

total

analysis only

no abstract acceleration

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/**x**/β	size	time	size	time	size	time
Gate 1	4/4/2	7	?	5	**0.73**	24	?
Escalator 1	5/4/2	12	**0.14 (0.04)**	9	0.49	22	?
Traffic 1	4/6/0	18	**0.14 (0.01)**	16	0.19	5	3.49
Traffic 2	4/8/0	18	?	16	**0.35**	28	?
LCM Quest 0a-1	7/2/0	7	**0.04 (0.01)**	5	**0.04**	5	0.05
LCM Quest 0a-2	7/3/0	6	**0.05 (0.01)**	4	**0.05**	8	0.19
LCM Quest 0b-1	10/3/0	19	**0.08 (0.01)**	12	**0.08**	9	?
LCM Quest 0b-2	10/4/0	17	**0.09 (0.01)**	11	0.20	33	?
LCM Quest 0c-1	15/4/0	28	0.17 (0.01)	16	**0.16**	8	0.86
LCM Quest 0c-2	15/5/0	25	**0.20 (0.05)**	14	0.24	50	14.8
LCM Quest 1b-1	16/5/0	55	0.92 (0.04)	29	**0.37**	15	?
LCM Quest 1b-2	16/5/0	45	0.76 (0.12)	23	**0.47**	61	?
LCM Quest 1-1	16/5/0	114	1.99 (0.48)	42	**0.92**	6	2.45
LCM Quest 1-2	16/6/0	100	?	34	?	>156	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation (cont)

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/**x**/β	size	time	size	time	size	time
LCM Quest 2-1	17/6/0	247	c	82	**7.84**	9	12.8
LCM Quest 2-2	17/7/0	198	>	62	?	>76	>
LCM Quest 3d-1	26/6/0	281	>	81	**5.43**	49	?
LCM Quest 3d-2	26/7/0	266	c	73	?	446	?
LCM Quest 3-1	25/5/0	483	26.5 (14.4)	58	8.49	12	**3.76**
LCM Quest 3-2	25/6/0	481	c	54	?	>1173	>
LCM Quest 3e-1	27/7/0	638	>	140	**20.6**	49	?
LCM Quest 3e-2	27/8/0	514	>	110	**6.46**	>28	>
LCM Quest 3c-1	26/6/0	1319	>	130	**34.2**	9	?
LCM Quest 3c-2	26/7/0	1056	c	98	?	>70	>
LCM Quest 3b-1	26/6/0	1724	>	170	43.8	14	**19.1**
LCM Quest 3b-2	26/7/0	1710	>	162	?	>32	>
LCM Quest 4-1	27/7/0	4482	>	386	186	9	**50.1**
LCM Quest 4-2	27/8/0	3586	>	290	?	>6	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation (cont)

state space explosion

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/x/β	size	time	size	time	size	time
LCM Quest 2-1	17/6/0	247	c	82	**7.84**	9	12.8
LCM Quest 2-2	17/7/0	198	>	62	?	>76	>
LCM Quest 3d-1	26/6/0	281	>	81	**5.43**	49	?
LCM Quest 3d-2	26/7/0	266	c	73	?	446	?
LCM Quest 3-1	25/5/0	483	26.5 (14.4)	58	8.49	12	**3.76**
LCM Quest 3-2	25/6/0	481	c	54	?	>1173	>
LCM Quest 3e-1	27/7/0	638	>	140	**20.6**	49	?
LCM Quest 3e-2	27/8/0	514	>	110	**6.46**	>28	>
LCM Quest 3c-1	26/6/0	1319	>	130	**34.2**	9	?
LCM Quest 3c-2	26/7/0	1056	c	98	?	>70	>
LCM Quest 3b-1	26/6/0	1724	>	170	43.8	14	**19.1**
LCM Quest 3b-2	26/7/0	1710	>	162	?	>32	>
LCM Quest 4-1	27/7/0	4482	>	386	186	9	**50.1**
LCM Quest 4-2	27/8/0	3586	>	290	?	>6	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation (cont)

state space explosion

CFG 10 times smaller

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/**x**/β	size	time	size	time	size	time
LCM Quest 2-1	17/6/0	247	c	82	**7.84**	9	12.8
LCM Quest 2-2	17/7/0	198	>	62	?	>76	>
LCM Quest 3d-1	26/6/0	281	>	81	**5.43**	49	?
LCM Quest 3d-2	26/7/0	266	c	73	?	446	?
LCM Quest 3-1	25/5/0	483	26.5 (14.4)	58	8.49	12	**3.76**
LCM Quest 3-2	25/6/0	481	c	54	?	>1173	>
LCM Quest 3e-1	27/7/0	638	>	140	**20.6**	49	?
LCM Quest 3e-2	27/8/0	514	>	110	**6.46**	>28	>
LCM Quest 3c-1	26/6/0	1319	>	130	**34.2**	9	?
LCM Quest 3c-2	26/7/0	1056	c	98	?	>70	>
LCM Quest 3b-1	26/6/0	1724	>	170	43.8	14	**19.1**
LCM Quest 3b-2	26/7/0	1710	>	162	?	>32	>
LCM Quest 4-1	27/7/0	4482	>	386	186	9	**50.1**
LCM Quest 4-2	27/8/0	3586	>	290	?	>6	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Experimental Evaluation (cont)

state space explosion

CFG 10 times smaller

incremental partitioning

	vars	ASPIC (Gonnord 2009)		NBACCEL (this talk)		NBAC (Jeannet 2003)	
	b/**x**/β	size	time	size	time	size	time
LCM Quest 2-1	17/6/0	247	c	82	**7.84**	9	12.8
LCM Quest 2-2	17/7/0	198	>	62	?	>76	>
LCM Quest 3d-1	26/6/0	281	>	81	**5.43**	49	?
LCM Quest 3d-2	26/7/0	266	c	73	?	446	?
LCM Quest 3-1	25/5/0	483	26.5 (14.4)	58	8.49	12	**3.76**
LCM Quest 3-2	25/6/0	481	c	54	?	>1173	>
LCM Quest 3e-1	27/7/0	638	>	140	**20.6**	49	?
LCM Quest 3e-2	27/8/0	514	>	110	**6.46**	>28	>
LCM Quest 3c-1	26/6/0	1319	>	130	**34.2**	9	?
LCM Quest 3c-2	26/7/0	1056	c	98	?	>70	>
LCM Quest 3b-1	26/6/0	1724	>	170	43.8	14	**19.1**
LCM Quest 3b-2	26/7/0	1710	>	162	?	>32	>
LCM Quest 4-1	27/7/0	4482	>	386	186	9	**50.1**
LCM Quest 4-2	27/8/0	3586	>	290	?	>6	>

?	"don´t know" (property not proved)
>	timed out after 600s
c	out of memory or crashed

Outline

Alternative Methods for Logico-Numerical Programs

- Bultan et al 2000:
  - Model-checking of logico-numerical programs using BDDs and Presburger formulae
  - Neither acceleraton nor widening
- Fränzle and Herde 2007 (HySat):
  - Bounded model-checking for hybrid systems using constraint solving
  - Boolean control structure encoded in linear pseudo-Boolean constraints
- Hagen and Tinelli 2008 (Kind):
  - k-induction-based verification of data-flow programs using SMT-solvers

Conclusion

- Logico-numerical abstract acceleration method
  - based on partial decoupling of Boolean and numerical transitions
- Dedicated partitioning technique
  - that makes logico-numerical abstract acceleration effective

	abstract interpretation	abstract acceleration
enumeration of Boolean states	*classical AI*	ASPIC
state space partitioning	NBAC	NBACCEL

Results:

- Improved precision and performance
- Properties proved on much smaller CFGs

Future Work:

- Incremental partition refinement
- Partitioning by numerical constraints

Logico-Numerical Abstract Acceleration and Application to the Verification of Data-Flow Programs

Peter Schrammel and Bertrand Jeannet
{peter.schrammel,bertrand.jeannet}@inria.fr

INRIA Grenoble – Rhône-Alpes

The 18th International Static Analysis Symposium
September 14-16, 2011, Venice, Italy