Knowledge Compilation for Online Decision-Making: Application to the Control of Autonomous Systems

Alexandre Niveau
Hélène Fargier Cédric Pralet Gérard Verfaillie

March 27th, 2012

Autonomous System

- Autonomous system: must be able to make decisions, depending on the current situation
  - current observations
  - current goals

Example (Explorer robot)

- explores a zone
- gather samples
- when enough samples have been gathered, go to another zone

→ finding a decision policy

- Problem hard to solve in the general case

Control of an Autonomous System

- Solving the problem online:
  - limited by the embedded computational power

  → reactivity not ensured

- Solving the problem offline:
  - anticipate all decisions for every possible situation
  - embed a set of “decision rules”

  → ensures a good reactivity... but limited by the embedded memory space

Looking for a Compromise

- Need for a tradeoff between reactivity and spatial compactness

$\rightarrow$ maximizing reactivity under memory space constraints

- This is the object of knowledge compilation

Principle of Knowledge Compilation

- Idea: transforming the problem into a compiled form that
  - makes its resolution tractable
  - is as compact as possible
- Can be seen as a translation of the problem into some target compilation language.
- The choice of the target language is crucial

Principle of Knowledge Compilation

- Translation step: may be hard
  - $\rightarrow$ but done offline
  - $\rightarrow$ and done only once
- Resolution step: tractable
  - $\rightarrow$ fast even online
  - $\rightarrow$ and done countless times

$\rightarrow$ Knowledge compilation shifts as much as possible of the computational effort before the system's launching

Goal of the Thesis

- There exists numerous target compilation languages:
  - the decision diagrams family (BDDs, FBDDs, OBDDs. . .);
  - finite-state automata (MDDs);
  - the NNF family (DNNFs, d-DNNFs. . .). . .

Goal of the Thesis

- There exists numerous target compilation languages:
  - the decision diagrams family (BDDs, FBDDs, OBDDs...);
  - finite-state automata (MDDs);
  - the NNF family (DNNFs, d-DNNFs...)...
- Proven useful in various domains:
  - model-checking
  - product configuration
  - diagnostic
  - planning

Goal of the Thesis

- There exists numerous target compilation languages:
  - the decision diagrams family (BDDs, FBDDs, OBDDs. . .);
  - finite-state automata (MDDs);
  - the NNF family (DNNFs, d-DNNFs. . .). . .
- Proven useful in various domains:
  - model-checking
  - product configuration
  - diagnostic
  - planning

Goal of the thesis

Study whether KC can be applied to realistic problems of aeronautical or spatial autonomous system control.

Outline

Outline

Outline

Planning Problem

- Framework: non-deterministic planning
  - state variables, decision variables
  - initial states, goal states
  - transition relation

Example (Transition relation)

current state	**decision**	**next state**
$light$,	$press\_button$	$\neg light$, $\neg button$
$\neg light$, $\neg button$, $bulb\_OK$	$press\_button$	$light$, $button$, $bulb\_OK$
$\neg light$, $\neg button$, $bulb\_OK$	$press\_button$	$\neg light$, $button$, $\neg bulb\_OK$
$\neg bulb\_OK$	$change\_bulb$	$bulb\_OK$
...	...	...

Decision Policy

- Solution: decision policy (relation associating actions to each reachable state)

Example (Decision policy)

if you observe that...	**then...**
there is no light and the button is off	press the button
there is no light and the button is on	change the bulb
there is light	enjoy

$\rightarrow$ Boolean function δ, involving two kinds of variables:

- state variables S;
- decision variables D.

- $\delta(\vec{s}, \vec{d}) = \top \quad \rightarrow \quad \vec{d}$ is a suitable decision in state $\vec{s}$.

Decision Policy as a Boolean Function

Example (Decision policy)

if. . .	**then. . .**
there is no light and the button is off	press the button
there is no light and the button is on	change the bulb
there is light	enjoy

Decision Policy as a Boolean Function

Example (Decision policy)

$\vec{s}$	$\vec{d}$
$\neg light,\ \neg button$	$press\_button,\ \neg change\_bulb$
$\neg light,\ button$	$\neg press\_button,\ change\_bulb$
$light,\ button$	$\neg press\_button,\ \neg change\_bulb$
$light,\ \neg button$	$\neg press\_button,\ \neg change\_bulb$

Decision Policy as a Boolean Function

Example (Decision policy)

$\vec{s}$	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
$\neg light$, $\neg button$	$press\_button$, $\neg change\_bulb$	$\top$
$\neg light$, $button$	$\neg press\_button$, $change\_bulb$	$\top$
$light$, $button$	$\neg press\_button$, $\neg change\_bulb$	$\top$
$light$, $\neg button$	$\neg press\_button$, $\neg change\_bulb$	$\top$
$\neg light$, $\neg button$	$\neg press\_button$, $\neg change\_bulb$	$\bot$
$\neg light$, $\neg button$	$\neg press\_button$, $change\_bulb$	$\bot$
$\neg light$, $button$	$\neg press\_button$, $\neg change\_bulb$	$\bot$
$light$, $\neg button$	$press\_button$, $\neg change\_bulb$	$\bot$
...	...	$\bot$
...	...	$\bot$

Exploiting a Policy

Two operations needed:

- assign state variables according to observations: conditioning

Example (Exploiting a policy)

$\vec{s}$	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
$\neg light, \neg button$	$press\_button, \neg change\_bulb$	$\top$
$\neg light, button$	$\neg press\_button, change\_bulb$	$\top$
$light, button$	$\neg press\_button, \neg change\_bulb$	$\top$
$light, \neg button$	$\neg press\_button, \neg change\_bulb$	$\top$
$\neg light, \neg button$	$\neg press\_button, \neg change\_bulb$	$\bot$
$\neg light, \neg button$	$\neg press\_button, change\_bulb$	$\bot$
$\neg light, button$	$\neg press\_button, \neg change\_bulb$	$\bot$
$light, \neg button$	$press\_button, \neg change\_bulb$	$\bot$
...	...	$\bot$

Exploiting a Policy

Two operations needed:

- assign state variables according to observations: conditioning

Example (Exploiting a policy)

$\vec{s}$	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
$\neg light, \neg button$	$press\_button, \neg change\_bulb$	$\top$
$\neg light, button$	$\neg press\_button, change\_bulb$	$\top$
$light, button$	$\neg press\_button, \neg change\_bulb$	$\top$
$light, \neg button$	$\neg press\_button, \neg change\_bulb$	$\top$
$\neg light, \neg button$	$\neg press\_button, \neg change\_bulb$	$\bot$
$\neg light, \neg button$	$\neg press\_button, change\_bulb$	$\bot$
$\neg light, button$	$\neg press\_button, \neg change\_bulb$	$\bot$
$light, \neg button$	$press\_button, \neg change\_bulb$	$\bot$
...	...	$\bot$

Exploiting a Policy

Two operations needed:

- assign state variables according to observations: conditioning

Example (Exploiting a policy)

$\vec{s}$	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
$\neg light$, $\neg button$	$press\_button$, $\neg change\_bulb$	$\top$
$\neg light$, $button$	$\neg press\_button$, $change\_bulb$	$\top$
$light$, $button$	$\neg press\_button$, $\neg change\_bulb$	$\top$
$light$, $\neg button$	$\neg press\_button$, $\neg change\_bulb$	$\top$
$\neg light$, $\neg button$	$\neg press\_button$, $\neg change\_bulb$	$\bot$
$\neg light$, $\neg button$	$\neg press\_button$, $change\_bulb$	$\bot$
$\neg light$, $button$	$\neg press\_button$, $\neg change\_bulb$	$\bot$
$light$, $\neg button$	$press\_button$, $\neg change\_bulb$	$\bot$
...	...	$\bot$

Exploiting a Policy

Two operations needed:

- assign state variables according to observations: conditioning

Example (Exploiting a policy)

	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
	$press\_button, \neg change\_bulb$	$\top$
	$\neg press\_button, change\_bulb$	$\top$
	$\neg press\_button, \neg change\_bulb$	$\top$
	$\neg press\_button, \neg change\_bulb$	$\top$
	$\neg press\_button, \neg change\_bulb$	$\bot$
	$\neg press\_button, change\_bulb$	$\bot$
	$\neg press\_button, \neg change\_bulb$	$\bot$
	$press\_button, \neg change\_bulb$	$\bot$
	...	$\bot$

Exploiting a Policy

Two operations needed:

- assign state variables according to observations: conditioning
- produce one action among the possible ones: model extraction

Example (Exploiting a policy)

	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
	$press\_button, \neg change\_bulb$	$\top$
	$\neg press\_button, \neg change\_bulb$	$\perp$
	$\neg press\_button, change\_bulb$	$\perp$

Exploiting a Policy

Two operations needed:

- assign state variables according to observations: conditioning
- produce one action among the possible ones: model extraction

Example (Exploiting a policy)

	$\vec{d}$	$\delta(\vec{s}, \vec{d})$
	press_button, $\neg$*change_bulb*	$\top$
	$\neg$*press_button*, $\neg$*change_bulb*	$\bot$
	$\neg$*press_button*, *change_bulb*	$\bot$

Outline

Knowledge Compilation

- A problem is a set of operations on a knowledge base
  - knowledge base: propositional formula, constraint network, . . .
  - operations: combining formulas, checking properties, . . .
- Compilation of a problem: translation of the knowledge base into a target language such that
  - operations on the compiled form are tractable
  - the compiled form is as compact as possible

Knowledge Compilation for Planning

Applying KC to the control of autonomous systems:

- planning problem solved online
  - compilation of a transition relation
  - operations on the compiled form: conditioning, variable elimination. . .

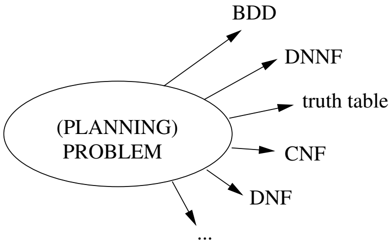
Knowledge Compilation for Planning

Applying KC to the control of autonomous systems:

- planning problem solved online
  - compilation of a transition relation
  - operations on the compiled form: conditioning, variable elimination. . .
- planning problem solved offline
  - compilation of a decision policy
  - operations on the compiled form: conditioning, model extraction

Target Languages

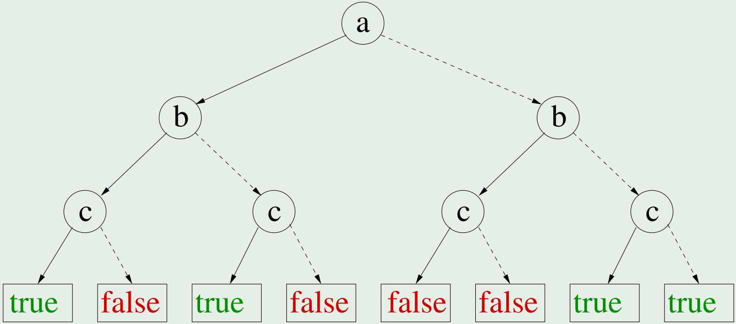
- Various target compilation languages are suitable



- An example: binary decision diagrams (BDDs)

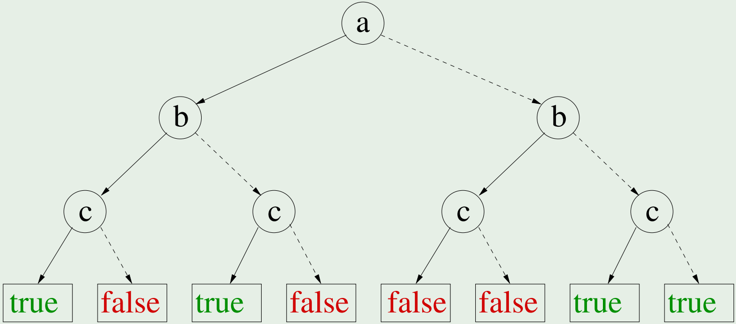
Compilation: Example of BDDs

Example : $(b \to a) \land (a \to c)$



Compilation: Example of BDDs

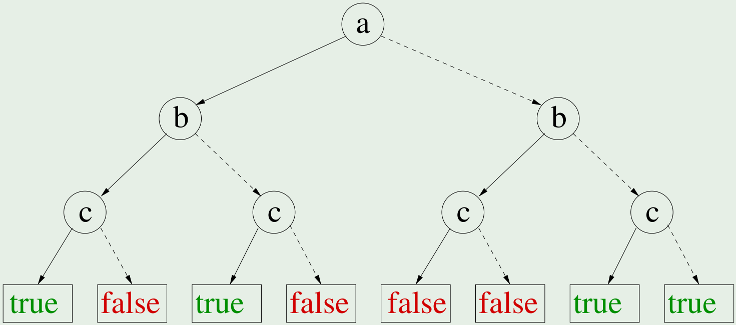
Example : $(b \rightarrow a) \wedge (a \rightarrow c)$



- Each path in the tree: assignment of all variables.
- The leaf is the value of the function for this assignment.

Compilation: Example of BDDs

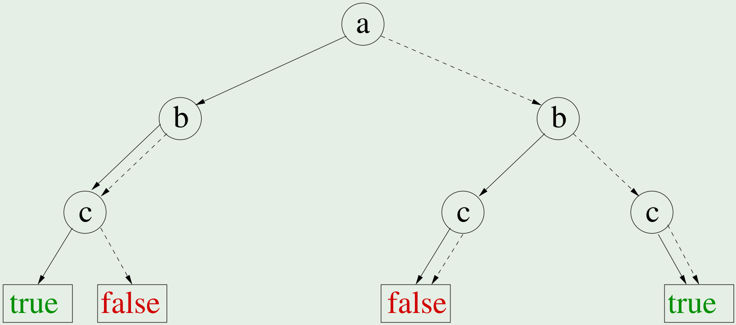
Example : $(b \rightarrow a) \wedge (a \rightarrow c)$



- Merging isomorphic subgraphs...

Compilation: Example of BDDs

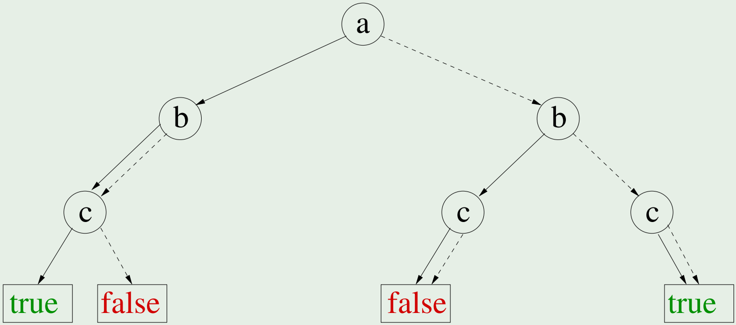
Example : $(b \rightarrow a) \wedge (a \rightarrow c)$



- Isomorphic subgraphs merged

Compilation: Example of BDDs

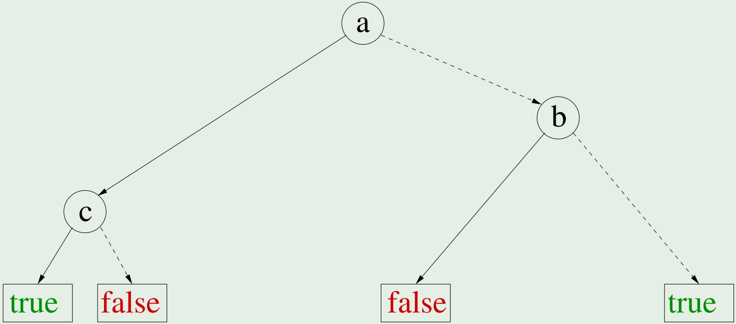
Example : $(b \rightarrow a) \wedge (a \rightarrow c)$



- Removing redundant nodes. . .

Compilation: Example of BDDs

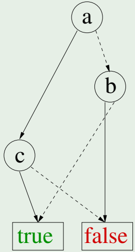
Example : $(b \to a) \wedge (a \to c)$



- Redundant nodes removed

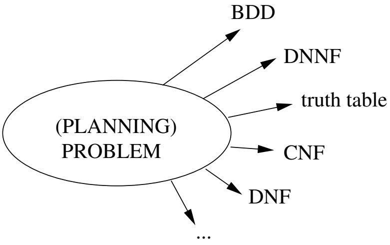
Compilation: Example of BDDs

Example : $(b \to a) \wedge (a \to c)$



- The BDD can be exponentially more compact than the tree

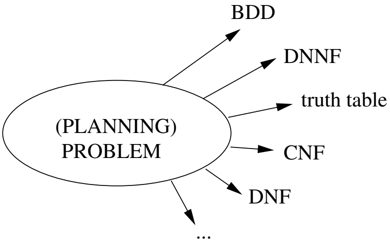
Choosing a Target Language



- What is the most appropriate for my application?

[Darwiche and Marquis, 2002] Darwiche, A. and Marquis, P. (2002). A Knowledge Compilation Map. *JAIR*, 17:229–264.

Choosing a Target Language



- What is the most appropriate for my application? → use the knowledge compilation map [Darwiche and Marquis, 2002]
- Compares target languages according to their:
  - efficiency on operations
  - succinctness.

[Darwiche and Marquis, 2002] Darwiche, A. and Marquis, P. (2002). A Knowledge Compilation Map. *JAIR*, 17:229–264.

Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary queries and transformations

L	CO (consistency)	VA (validity)	MX (model extr.)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○	○
DNNF	√	○	√	√	○	○	○	○	√
BDD	○	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	√	○	√	√
DNF	√	○	√	√	○	○	○	○	√
CNF	○	√	○	○	√	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	$\wedge$C (conjunction)	$\wedge$BC (bounded conj.)	$\vee$C (disjunction)	$\vee$BC (bounded disj.)	$\neg$C (negation)
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
BDD	√	○	√	√	√	√	√	√
FBDD	√	●	○	●	○	●	○	√
OBDD	√	●	√	●	○	●	○	√
DNF	√	√	√	●	√	√	√	●
CNF	√	○	√	√	√	●	√	●

√ polytime
○ not polytime unless P = NP
● not polytime

Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary queries and transformations

L	CO (consistency)	VA (validity)	MX (model extr.)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○	○
DNNF	√	○	√	√	○	○	○	○	√
BDD	○	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	√	○	√	√
DNF	√	○	√	√	○	○	○	○	√
CNF	○	√	○	○	√	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	$\wedge$C (conjunction)	$\wedge$BC (bounded conj.)	$\vee$C (disjunction)	$\vee$BC (bounded disj.)	$\neg$C (negation)
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
BDD	√	○	√	√	√	√	√	√
FBDD	√	●	○	●	○	●	○	√
OBDD	√	●	√	●	○	●	○	√
DNF	√	√	√	●	√	√	√	●
CNF	√	○	√	√	√	●	√	●

√ polytime
○ not polytime unless P = NP
● not polytime

Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary queries and transformations

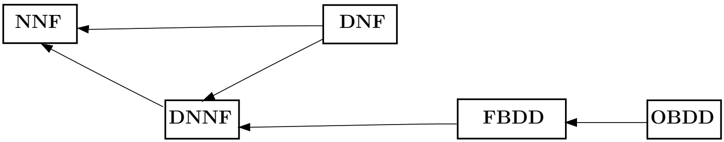
L	CO (consistency)	VA (validity)	MX (model extr.)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○	○
DNNF	√	○	√	√	○	○	○	○	√
BDD	○	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	√	○	√	√
DNF	√	○	√	√	○	○	○	○	√
CNF	○	√	○	○	√	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	∧C (conjunction)	∧BC (bounded conj.)	∨C (disjunction)	∨BC (bounded disj.)	¬C (negation)
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
BDD	√	○	√	√	√	√	√	√
FBDD	√	•	○	•	○	•	○	√
OBDD	√	•	√	•	○	•	○	√
DNF	√	√	√	•	√	√	√	•
CNF	√	○	√	√	√	•	√	•

√ polytime
○ not polytime unless P = NP
• not polytime

Knowledge Compilation Map: Succinctness

- Succinctness relation: orders target languages w.r.t. their compacity



- $L_1 \longleftarrow L_2$: L_1 is strictly more succinct than L_2

Target Languages for Planning Applications?

- Many target languages can be used for our application

$\rightarrow$ Boolean or enumerated variables only

- Real applications often involve continuous variables (time, energy. . .)

Our work

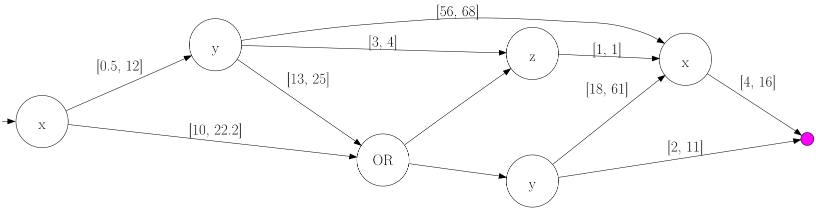
Define languages representing Boolean functions over variables with continuous or large enumerated domains.

Outline

Outline

Interval Automata

- The first language we described: interval automata (IAs).

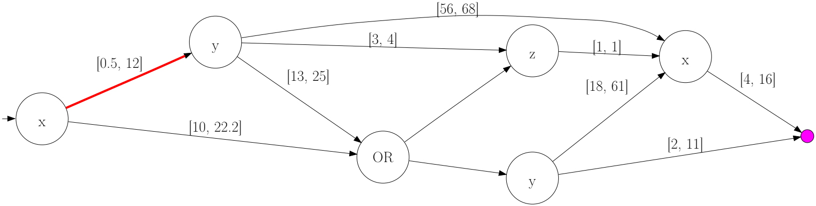


Definition (Interval automaton)

An interval automaton (IA) is a directed acyclic graph with at most one root and at most one leaf, with nodes labeled with a variable or with "OR", and edges labeled by a closed interval from $\mathbb{R}$.

Interval Automata

- The first language we described: interval automata (IAs).



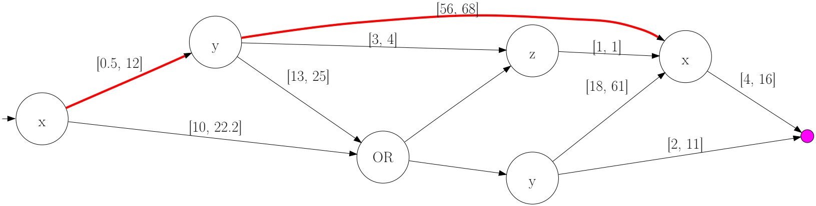
$x \in [0.5, 12]$

Definition (Interval automaton)

An interval automaton (IA) is a directed acyclic graph with at most one root and at most one leaf, with nodes labeled with a variable or with "OR", and edges labeled by a closed interval from $\mathbb{R}$.

Interval Automata

- The first language we described: interval automata (IAs).



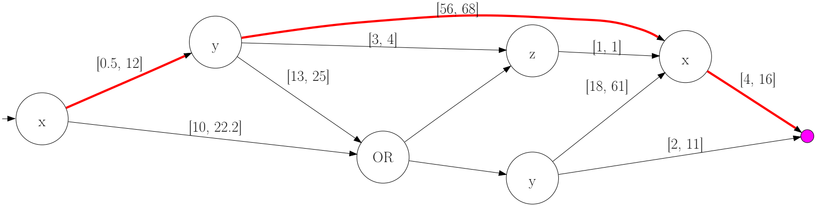
$x \in [0.5, 12] \qquad \times \qquad y \in [56, 68]$

Definition (Interval automaton)

An interval automaton (IA) is a directed acyclic graph with at most one root and at most one leaf, with nodes labeled with a variable or with "OR", and edges labeled by a closed interval from $\mathbb{R}$.

Interval Automata

- The first language we described: interval automata (IAs).



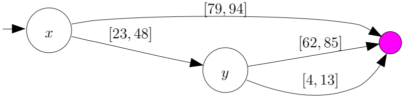
$$x \in [0.5, 12] \cap [4, 16] \quad \times \quad y \in [56, 68]$$

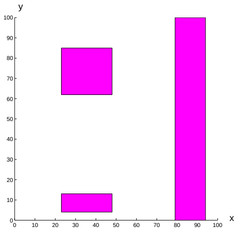
Definition (Interval automaton)

An interval automaton (IA) is a directed acyclic graph with at most one root and at most one leaf, with nodes labeled with a variable or with "OR", and edges labeled by a closed interval from $\mathbb{R}$.

Semantics of Interval Automata

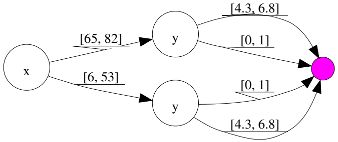
- Each interval automaton represents a Boolean function, or equivalently a set of solutions





Reduction: merging of isomorphic nodes

The size of IAs can be reduced thanks to several operations

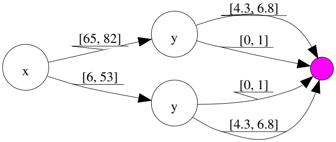


Isomorphic nodes

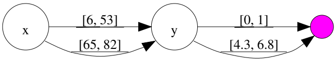
Two nodes N_1, N_2 of an IA φ are isomorphic iff $\text{Var}(N_1) = \text{Var}(N_2)$ and there exists a bijection σ from $\text{Out}(N_1)$ onto $\text{Out}(N_2)$, s.t. $\forall E \in \text{Out}(N_1), \text{Lbl}(E) = \text{Lbl}(\sigma(E))$ and $\text{Dest}(E) = \text{Dest}(\sigma(E))$.

Reduction: merging of isomorphic nodes

The size of IAs can be reduced thanks to several operations



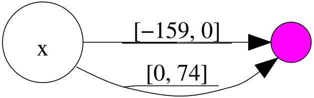
$\Downarrow$



Isomorphic nodes

Two nodes N_1, N_2 of an IA φ are isomorphic iff $\mathrm{Var}(N_1) = \mathrm{Var}(N_2)$ and there exists a bijection σ from $\mathrm{Out}(N_1)$ onto $\mathrm{Out}(N_2)$, s.t. $\forall E \in \mathrm{Out}(N_1), \mathrm{Lbl}(E) = \mathrm{Lbl}(\sigma(E))$ and $\mathrm{Dest}(E) = \mathrm{Dest}(\sigma(E))$.

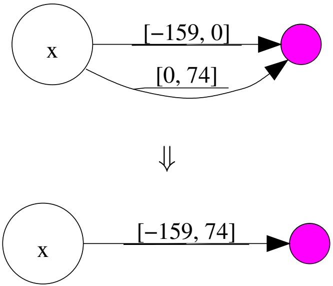
Reduction: merging of contiguous edges



Contiguous edges

Two edges E_1, E_2 of an IA φ are contiguous iff $\mathrm{Src}(E_1) = \mathrm{Src}(E_2)$, $\mathrm{Dest}(E_1) = \mathrm{Dest}(E_2)$ and there exists an interval $I \subseteq \mathbb{R}$ s.t. $\mathrm{Lbl}(E_1) \cup \mathrm{Lbl}(E_2) = I \cap \mathrm{Dom}(\mathrm{Var}(E_1))$.

Reduction: merging of contiguous edges

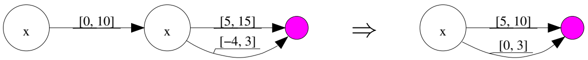


Contiguous edges

Two edges E_1, E_2 of an IA φ are contiguous iff $\mathrm{Src}(E_1) = \mathrm{Src}(E_2)$, $\mathrm{Dest}(E_1) = \mathrm{Dest}(E_2)$ and there exists an interval $I \subseteq \mathbb{R}$ s.t. $\mathrm{Lbl}(E_1) \cup \mathrm{Lbl}(E_2) = I \cap \mathrm{Dom}(\mathrm{Var}(E_1))$.

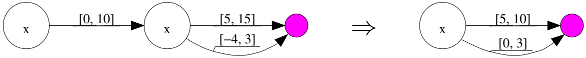
Reduction of an IA

Merging of stammering nodes:

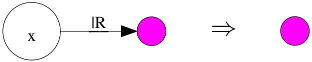


Reduction of an IA

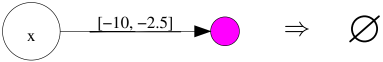
Merging of stammering nodes:



Elimination of undecisive nodes:

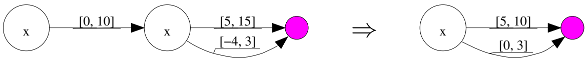


Elimination of unreachable edges (here $\mathrm{Dom}(x) = \mathbb{R}_+$):

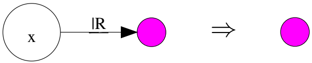


Reduction of an IA

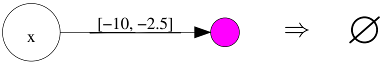
Merging of stammering nodes:



Elimination of undecisive nodes:



Elimination of unreachable edges (here $\mathrm{Dom}(x) = \mathbb{R}_+$):

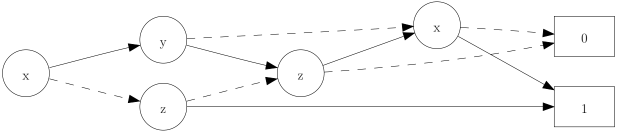


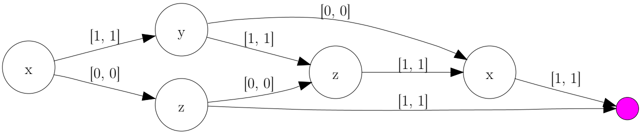
Theorem (Reduction of an IA)

There exists a polytime algorithm transforming any IA into an equivalent reduced IA.

Relation with BDD

- BDDs are particular IAs (Boolean variables, deterministic nodes).



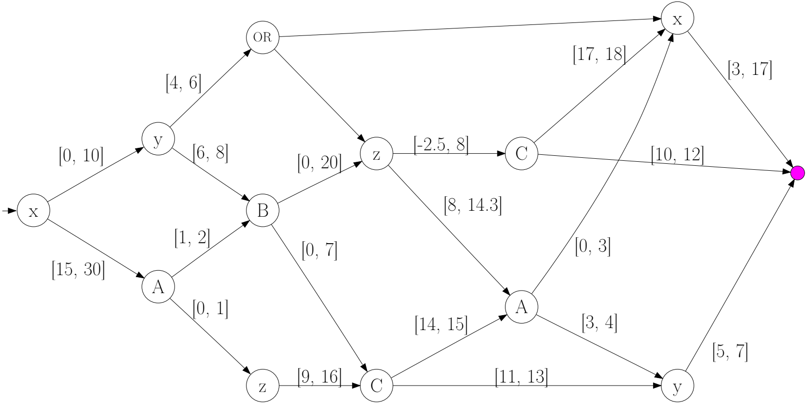


Outline

Conditioning an IA

We observe the state $x = 17, y = 6, z = 8$.

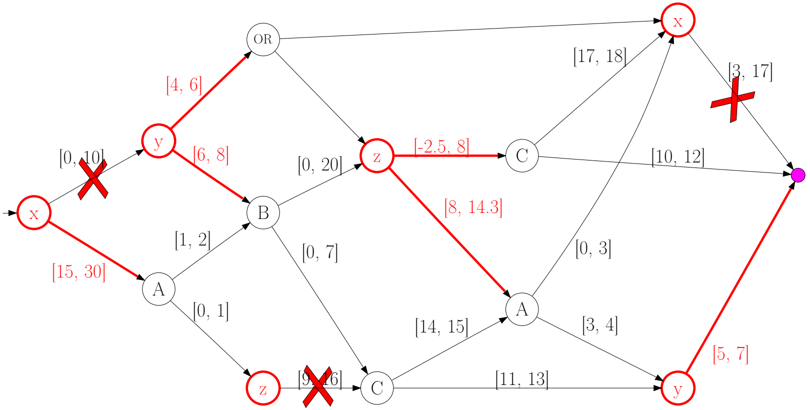
$\longrightarrow$ Conditioning of the policy :



Conditioning an IA

We observe the state $x = 17, y = 6, z = 8$.

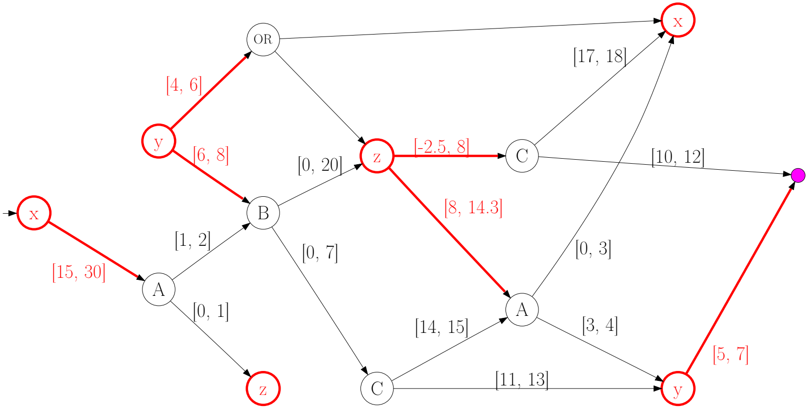
$\longrightarrow$ Conditioning of the policy :



Conditioning an IA

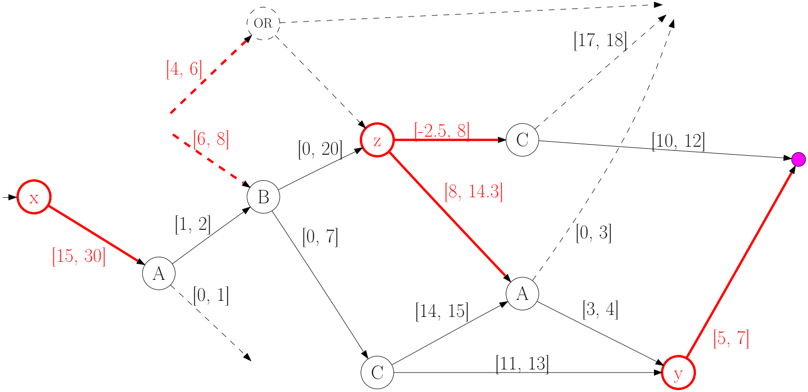
We observe the state $x = 17, y = 6, z = 8$.

$\longrightarrow$ Conditioning of the policy :



Conditioning an IA

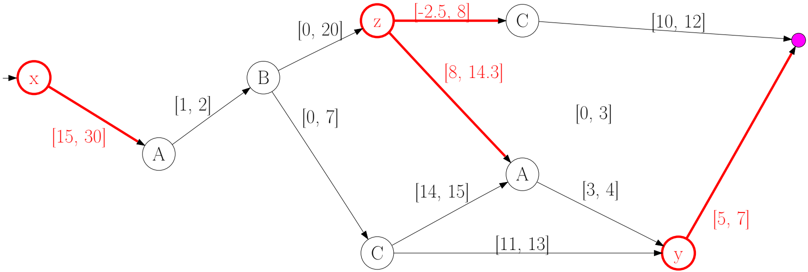
We observe the state $x = 17, y = 6, z = 8$.
$\longrightarrow$ Conditioning of the policy :



Conditioning an IA

We observe the state $x = 17, y = 6, z = 8$.

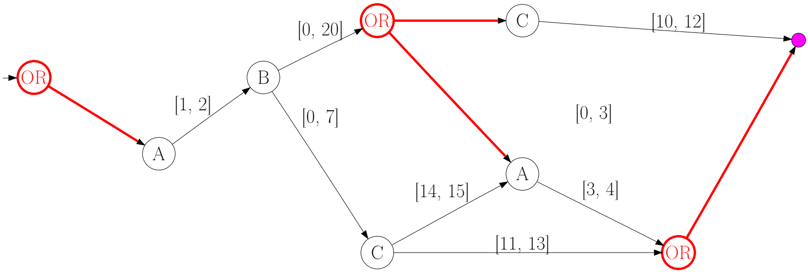
$\longrightarrow$ Conditioning of the policy :



Conditioning an IA

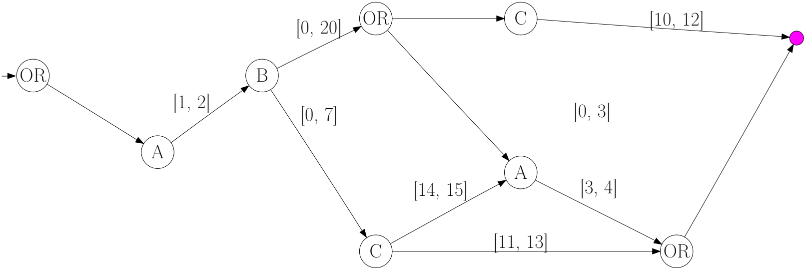
We observe the state $x = 17, y = 6, z = 8$.

$\longrightarrow$ Conditioning of the policy :



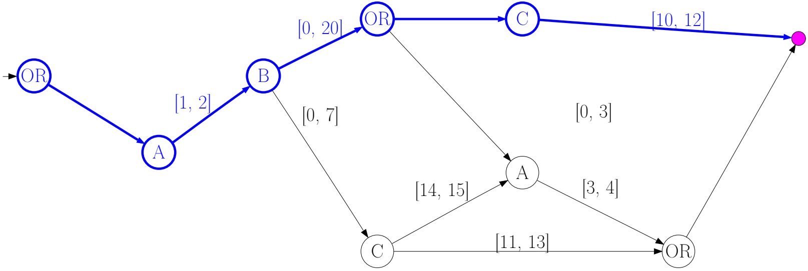
Model Extraction on an IA

We obtain a set of suitable decisions; we need to choose one
$\longrightarrow$ Model extraction



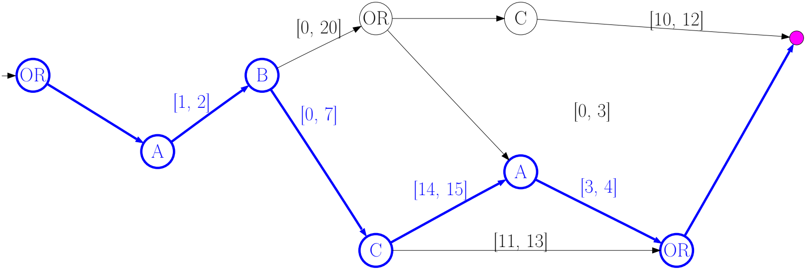
Model Extraction on an IA

We obtain a set of suitable decisions; we need to choose one
$\longrightarrow$ Model extraction



Model Extraction on an IA

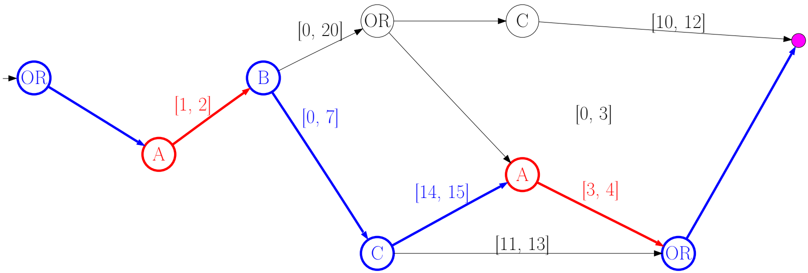
We obtain a set of suitable decisions; we need to choose one
$\longrightarrow$ Model extraction



Model Extraction on an IA

We obtain a set of suitable decisions; we need to choose one

$\longrightarrow$ Model extraction



Model Extraction is Hard

- Paths in an IA can be inconsistent

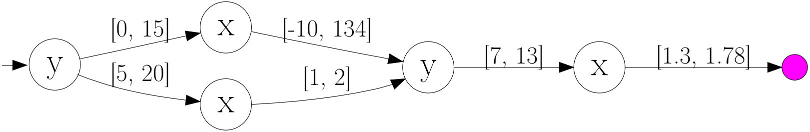
Theorem

Interval automata do not support model extraction in polytime.

$\rightarrow$ we look for a restriction on IAs, making this operation easier.

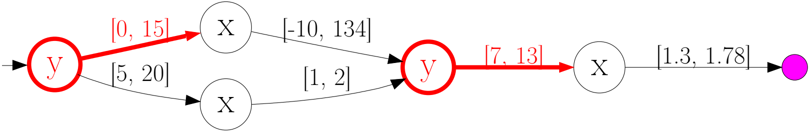
Focusing Interval Automata

- Idea: along a path, intervals can only shrink → focusing IAs



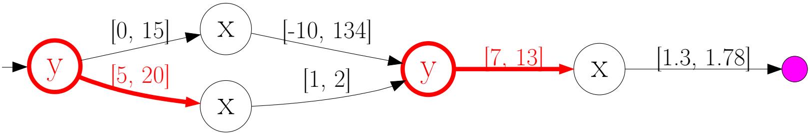
Focusing Interval Automata

- Idea: along a path, intervals can only shrink → focusing IAs



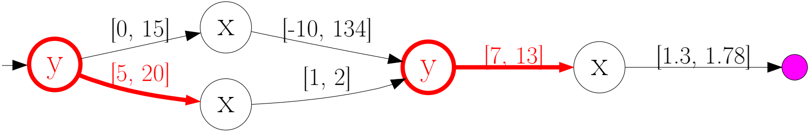
Focusing Interval Automata

- Idea: along a path, intervals can only shrink → focusing IAs



Focusing Interval Automata

- Idea: along a path, intervals can only shrink → focusing IAs



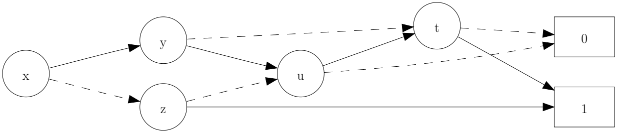
- Each path of a reduced FIA corresponds to at least one model

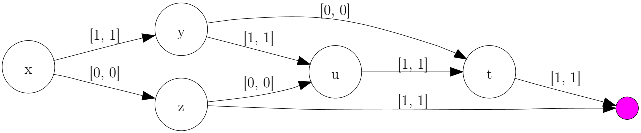
Theorem

FIAs support model extraction in polytime.

Read-once ⇒ Focusing

- Parallel with the "read-once" restriction on BDDs
- Read-once BDDs (FBDDs and OBDDs) are particular FIAs.



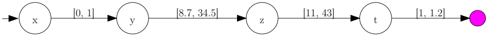


Outline

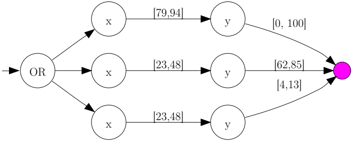
Compiling List of Boxes

- Compilation of decision policies, obtained with an external algorithm: list of "boxes" $\longmapsto$ FIA
  - Build the FIA representing each "box"

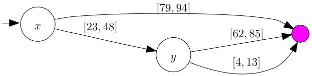
$$[0,1] \times [8.7, 34.5] \times [11, 43] \times [1, 1.2]$$



  - Make the disjunction of the boxes ($\vee$**C** easy on FIAs)



$\Rightarrow$

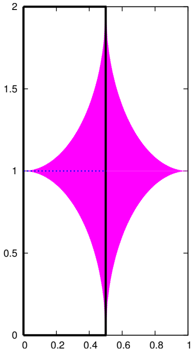


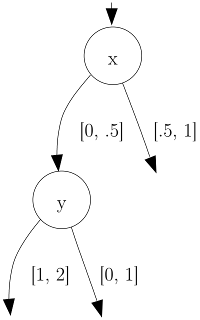
Compiling Continuous Constraint Networks

- Compilation of transition relations:

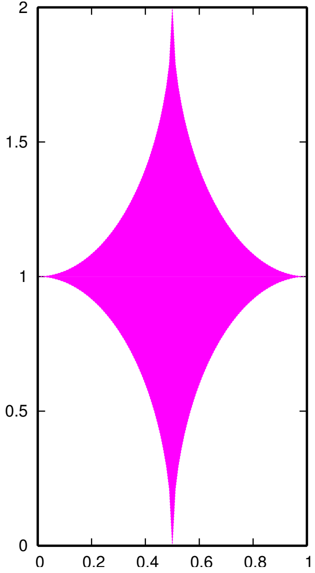
  continuous constraint networks $\longmapsto$ FIA

  - Following the "DPLL with a trace" approach: use the trace of the interval-based CSP solver RealPaver



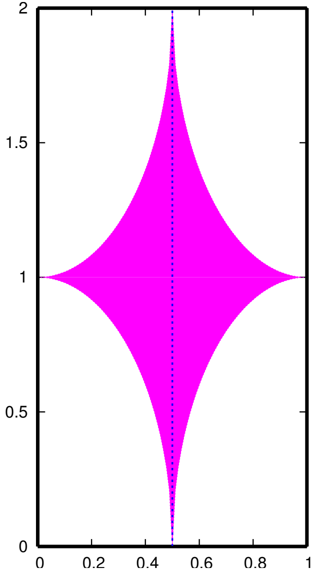


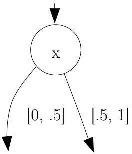
RealPaver with a Trace



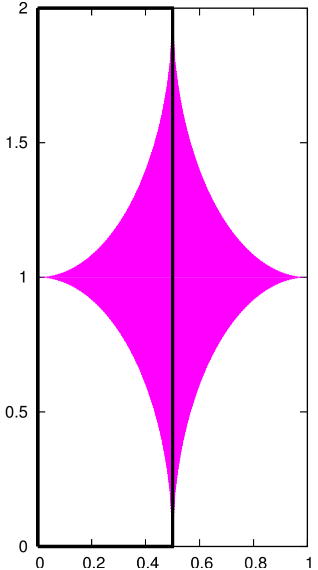
$$\begin{cases} y \geqslant 2\sqrt{\max(0, 0.25 - x^2)} \\ y \leqslant 2 - 2\sqrt{\max(0, 0.25 - x^2)} \\ y \geqslant 2\sqrt{\max(0, 0.25 - (x-1)^2)} \\ y \leqslant 2 - 2\sqrt{\max(0, 0.25 - (x-1)^2)} \end{cases}$$

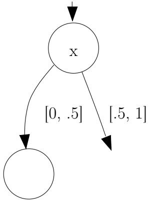
RealPaver with a Trace



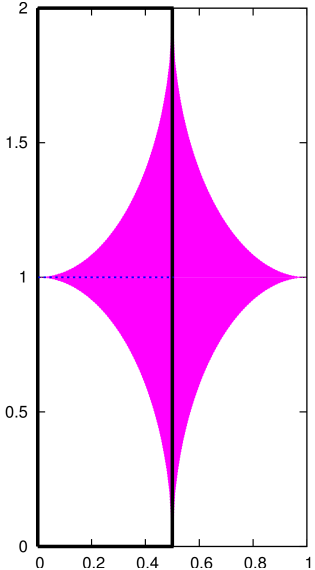


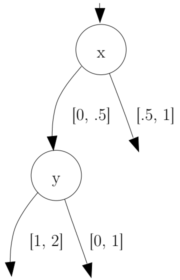
RealPaver with a Trace



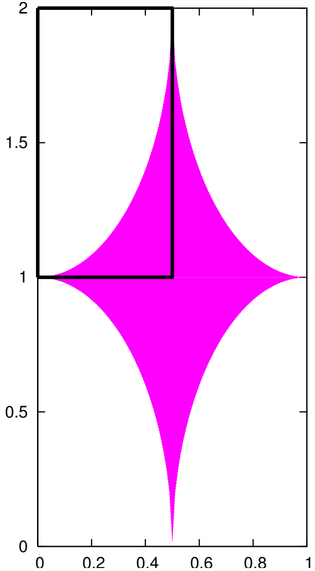


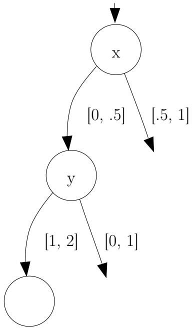
RealPaver with a Trace



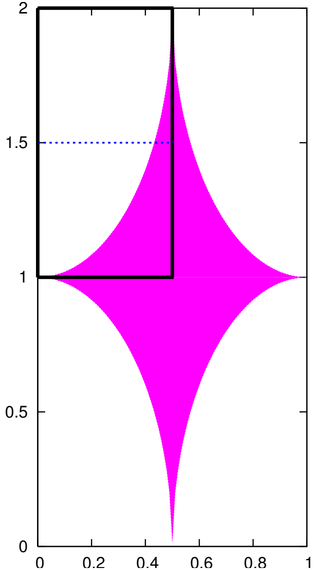


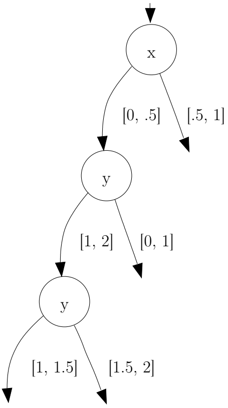
RealPaver with a Trace



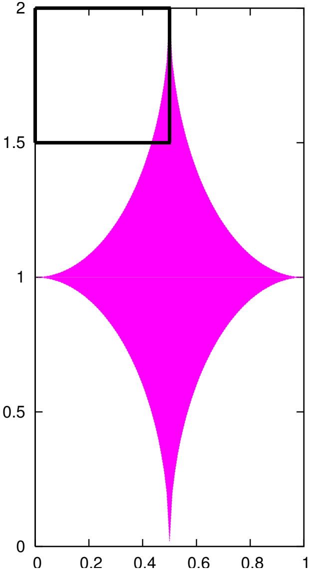


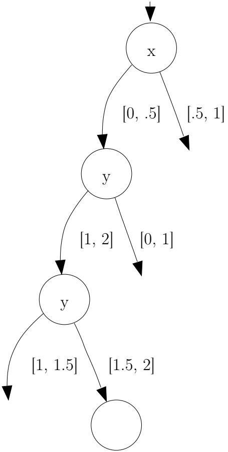
RealPaver with a Trace



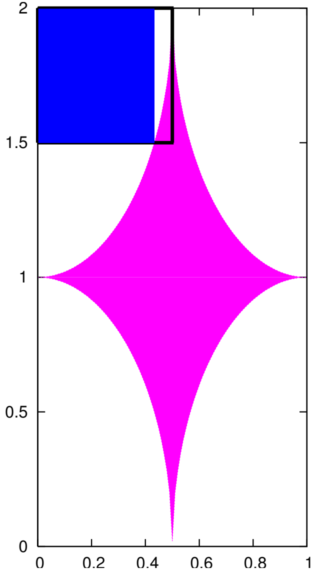


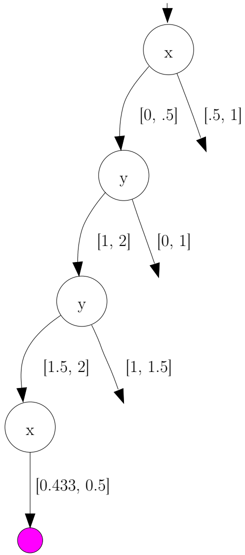
RealPaver with a Trace



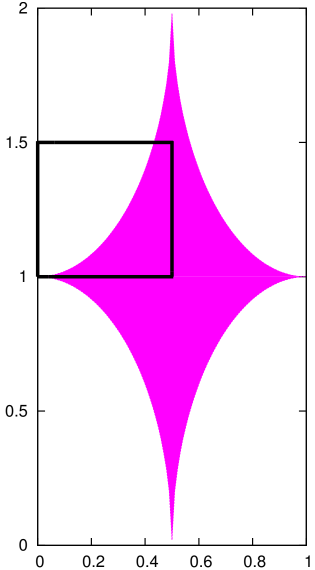


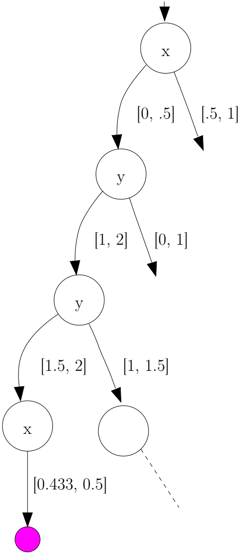
RealPaver with a Trace



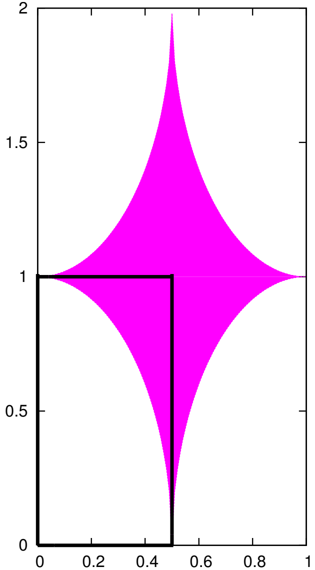


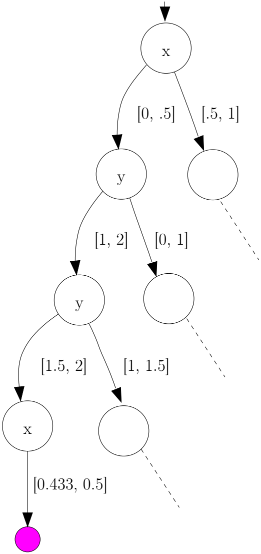
RealPaver with a Trace





RealPaver with a Trace





Implementation

- Prototype of "RealPaver with a trace"
- Toolbox for manipulating IAs/FIAs (**CO**, **MX**, **CD**, **FO**, $\wedge$**C**, ...)
- Experimenting exploitation of a transition table

problem	#edges	**CDFOMX**	**CDMX**	RealPaver
*Drone*4-5-3	61 596	<1 ms	1 ms	23 ms
*Drone*4-10-3	81 290	<1 ms	<1 ms	21 ms
*Drone*4-15-3	269 913	1 ms	1 ms	25 ms
*Drone*4-20-3	350 818	1 ms	1 ms	25 ms
*Drone*4-25-3	354 772	3 ms	3 ms	28 ms

$\rightarrow$ compatible with an online use

Knowledge Compilation Map of IAs and FIAs

Query	IA	FIA	DNNF
CO	∘	√	√
VA	∘	∘	∘
MC	√	√	√
CE	∘	√	√
IM	∘	∘	∘
EQ	∘	∘	∘
SE	∘	∘	∘
MX	∘	√	√
CX	∘	√	√
CT	∘	∘	∘
ME	∘	√	√

Transfo.	IA	FIA	DNNF
CD	√	√	√
TR	∘	√	√
FO	∘	√	√
SFO	√	√	√
EN	∘	∘	∘
SEN	√	∘	∘
∨C	√	√	√
∨BC	√	√	√
∨clC	√	√	√
∧C	√	∘	∘
∧BC	√	∘	∘
∧tC	√	√	√

√ polytime
∘ not polytime unless P = NP
• not polytime

Back to Enumerated Variables

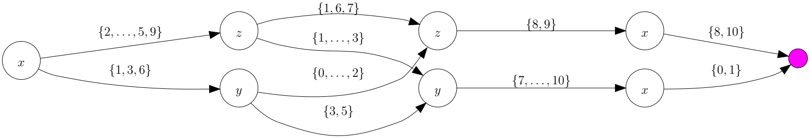
- Focusing IAs are not decomposable
- Yet the operations they support are similar to DNNF

→ We study how we can apply focusingness to languages on discrete variables

Outline

Set-labeled Diagrams

- We described a discrete counterpart to IAs: set-labeled diagrams (SDs)



- Edges of IAs labeled by intervals of real numbers
  $\rightarrow$ Edges of SDs labeled by sets of integers

The SD Family

- We considered a number of sublanguages of SDs, based on the following properties:
  - focusingness

    $\rightarrow$ FSDs
  - exclusive decision: imposes sets of sister edges to be disjoint

    $\rightarrow$ SDDs (set-labeled decision diagrams), FSDDs
  - fixed order on variables

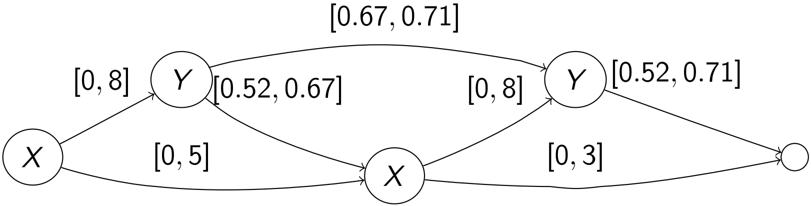
    $\rightarrow$ OSDs and OSDDs

Relationship with Other Languages

- MDDs are particular OSDDs (all sets are singletons)

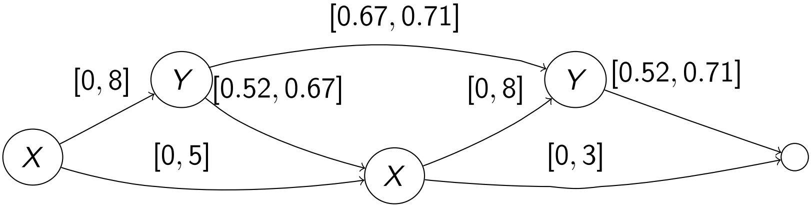
Relationship with Other Languages

- MDDs are particular OSDDs (all sets are singletons)
- FIAs can be "further compiled" into FSDs: judicious discretization



Relationship with Other Languages

- MDDs are particular OSDDs (all sets are singletons)
- FIAs can be "further compiled" into FSDs: judicious discretization

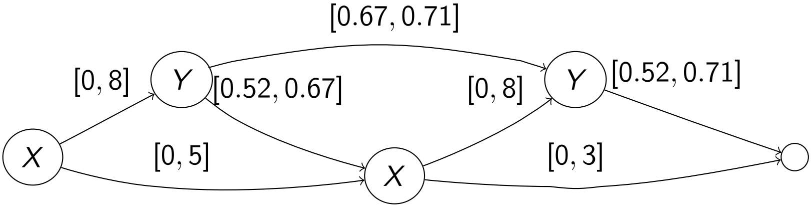


$$\{[0,3],\quad]3,5],\quad]5,8],\quad]8,10]\quad \}$$

$$\{[0,0.52[,\quad [0.52,0.67[,\quad \{0.67\},\quad]0.67,0.71]\quad]0.71,1]\quad \}$$

Relationship with Other Languages

- MDDs are particular OSDDs (all sets are singletons)
- FIAs can be "further compiled" into FSDs: judicious discretization

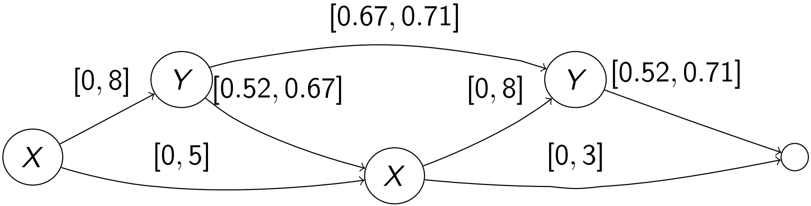


$$\{[0,3],\quad]3,5],\quad]5,8],\quad]8,10]\quad\}$$
$$\{x_1,\quad x_2,\quad x_3,\quad x_4\quad\}$$

$$\{[0,0.52[,\quad [0.52,0.67[,\quad \{0.67\},\quad]0.67,0.71]\quad]0.71,1]\quad\}$$

Relationship with Other Languages

- MDDs are particular OSDDs (all sets are singletons)
- FIAs can be "further compiled" into FSDs: judicious discretization

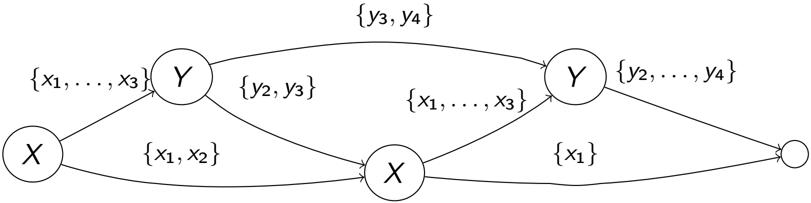


$$\begin{array}{lllll} \{[0,3], &]3,5], &]5,8], &]8,10] & \} \\ \{x_1, & x_2, & x_3, & x_4 & \} \end{array}$$

$$\begin{array}{llllll} \{[0,0.52[, & [0.52,0.67[, & \{0.67\}, &]0.67,0.71] &]0.71,1] & \} \\ \{y_1, & y_2, & y_3, & y_4, & y_5 & \} \end{array}$$

Relationship with Other Languages

- MDDs are particular OSDDs (all sets are singletons)
- FIAs can be "further compiled" into FSDs: judicious discretization



$$\begin{array}{llllll} \{[0,3], &]3,5], &]5,8], &]8,10] & & \} \\ \{x_1, & x_2, & x_3, & x_4 & & \} \end{array}$$

$$\begin{array}{llllll} \{[0,0.52[, & [0.52,0.67[, & \{0.67\}, &]0.67,0.71] &]0.71,1] & \} \\ \{y_1, & y_2, & y_3, & y_4, & y_5 & \} \end{array}$$

KC Map of the SD Family: Queries

Query	SD	SDD	FSD	FSDD	OSD	OSDD	$\text{OSD}_<$	$\text{OSDD}_<$	DNNF
CO	$\circ$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
VA	$\circ$	$\circ$	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$
MC	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
CE	$\circ$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
IM	$\circ$	$\circ$	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$
EQ	$\circ$	$\circ$	$\circ$	?	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$
SE	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\circ$
MX	$\circ$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
CX	$\circ$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
CT	$\circ$	$\circ$	$\circ$	?	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$
ME	$\circ$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

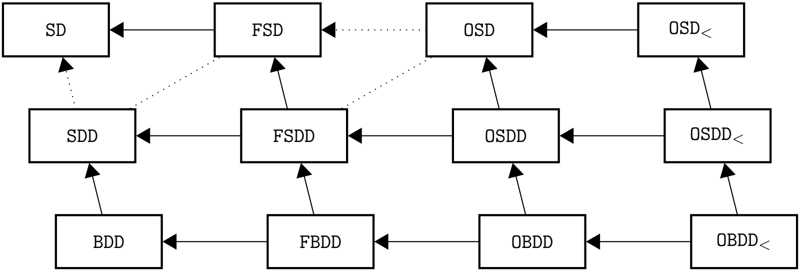
$\surd$ polytime
$\circ$ not polytime unless $P = NP$
$\bullet$ not polytime

KC Map of the SD Family: Transformations

Transfo.	SD	SDD	FSD	FSDD	OSD	OSDD	$\text{OSD}_<$	$\text{OSDD}_<$	DNNF
CD	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
TR	$\circ$	$\circ$	$\surd$	$\circ$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$
FO	$\circ$	$\circ$	$\surd$	$\circ$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$
SFO	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$
EN	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\circ$	$\bullet$	$\circ$
SEN	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\bullet$	$\circ$	$\bullet$	$\circ$
$\lor$**C**	$\surd$	$\surd$	$\surd$	$\circ$	?	$\bullet$	$\surd$	$\bullet$	$\surd$
$\lor$**BC**	$\surd$	$\surd$	$\surd$	$\circ$	?	$\circ$	$\surd$	$\surd$	$\surd$
$\lor$**clC**	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$\land$**C**	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\bullet$	$\circ$	$\bullet$	$\circ$
$\land$**BC**	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
$\land$**tC**	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$\neg$**C**	?	$\surd$	$\circ$	?	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$

$\surd$ polytime
$\circ$ not polytime unless $P = NP$
$\bullet$ not polytime

KC Map of the SD Family: Succinctness



- $\mathrm{L}_1 \longleftarrow \mathrm{L}_2$: L_1 is strictly more succinct than L_2
- $\mathrm{L}_1 \dashleftarrow \mathrm{L}_2$: unknown relationship

Choco with a Trace

- To compile discrete CSPs into FSDDs: "Choco with a trace"
- Different settings of Choco → different sublanguages
  - if variables are examined in fixed order → OSDDs
  - to get "pure" FSDDs, we can design heuristics for dynamic variable choice

Outline

Contributions

- Definition of interval automata and set-labeled diagrams
- Identification of the focusingness property
- KC map of IAs and FIAs / of the SD family
- "RealPaver/Choco with a trace" algorithms
- Implementation (compilers and toolboxes)

Perspectives

- Approximate compilation
- AND-nodes
- compilation modulo theory: more expressive literals
  $x \in [1, 4] \quad \longrightarrow \quad x - y < 3$
- Other data structures as compiled forms:
  $R^\star$-trees, hashtables...
- Study comparison of languages w.r.t. a representation change