Knowledge Compilation Using Interval Automata and Applications to Planning

Alexandre Niveau
Hélène Fargier Cédric Pralet Gérard Verfaillie

August 19th, 2010

Outline

Autonomous System Control

Control of autonomous systems: *decision-making tasks*, depending on the current observations and goals.

Decision-making has to be performed *online*, however, it is often combinatory.

Autonomous System Control

Control of autonomous systems: *decision-making tasks*, depending on the current observations and goals.

Decision-making has to be performed *online*, however, it is often combinatory.

- Performing these tasks *completely online*: compromise the *reactivity* of the system;
- Computing them *offline* (anticipating every possible situation): problematic regarding the limited size of embedded memory.

A possible way to solve this dilemma is to use *knowledge compilation*.

Principle of Knowledge Compilation

Knowledge compilation

- Idea: transforming the problem offline into a form that is tractable online.
- The problem is "translated" into a certain formalism (or *language*).
- This representation allows necessary operations to be tractable, while being as compact as possible.

Using compilation offline, one carries out computational parts before the system's setting up.

Approach

Framework: *strong non-deterministic planning*

We want to embed a complete solution, in the form of a *decision policy*, *i.e.* a relation associating actions to each reachable state.

Approach

Framework: *strong non-deterministic planning*

We want to embed a complete solution, in the form of a *decision policy*, *i.e.* a relation associating actions to each reachable state.

$\longrightarrow$ Boolean function, involving two kinds of variables:

- *state* variables S;
- *decision* variables D.

For some given $\vec{s}$ and $\vec{d}$, the function returns "true" iff decision $\vec{d}$ is convenient in state $\vec{s}$.

Goal of the paper

Many compilation structures could be used this way:

- finite-state automata;
- the binary decision diagrams family (BDDs, FBDDs, OBDDs. . .);
- the NNF family (DNNFs, d-DNNFs. . .). . .

$\longrightarrow$ Boolean or enumerated variables only.

Goal of the paper

Many compilation structures could be used this way:

- finite-state automata;
- the binary decision diagrams family (BDDs, FBDDs, OBDDs. . .);
- the NNF family (DNNFs, d-DNNFs. . .). . .

$\longrightarrow$ Boolean or enumerated variables only.

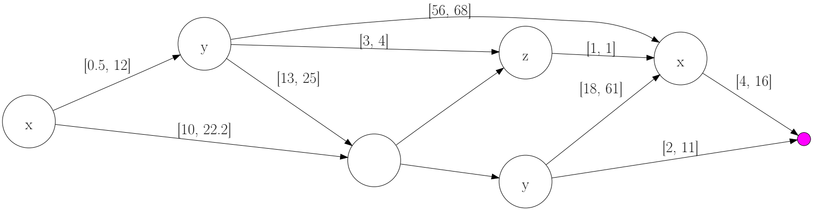
However, continuous variables involved in many real applications (time, energy. . .).
Goal: define new structures representing Boolean functions bearing on continuous (or large enumerated domain) variables.

Outline

Outline

Interval Automata

We have described the interval automata language (IA).
Hypothesis: variable domains are *finite* union of closed intervals from $\mathbb{R}$.

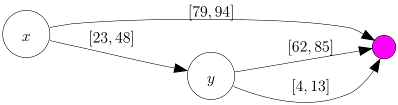


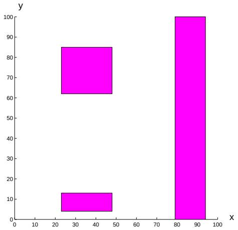
Interval automaton

An *interval automaton* (IA) is a couple $\phi = \langle X, \Gamma \rangle$, with

- X a finite and totally ordered set of variables;
- Γ a directed acyclic graph with at most one root and at most one leaf (the *sink*), whose non-leaf nodes are labelled by a variable of X or by the disjunctive symbol $\vee$, and whose edges are labelled by a closed interval from $\mathbb{R}$.

Semantics of Interval Automata





Each interval automaton (IA) represents a boolean function over the considered variables, or equivalently a set of solutions.

Interpretation function of an interval automaton

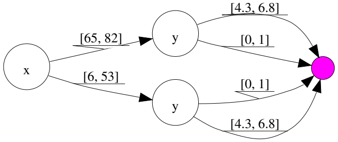
The *interpretation function* of an interval automaton ϕ on X is the function $\mathsf{I}(\phi)$ from $\mathsf{Dom}(X)$ to $\{\top, \bot\}$ defined as follows: for every X-assignment $\vec{x}$, $\mathsf{I}(\phi)(\vec{x}) = \top$ if and only if there exists a path p from the root to the sink of ϕ such that for each edge E along p, either $\mathsf{Var}(E) = \vee$ and $\mathsf{Itv}(E) \neq \emptyset$, or $\vec{x}(\mathsf{Var}(E)) \in \mathsf{Itv}(E)$.

Reduction of an IA: merging of isomorphic nodes

We described several operations allowing to reduce an IA's size without changing its semantics.

Reduction of an IA: merging of isomorphic nodes

We described several operations allowing to reduce an IA's size without changing its semantics.

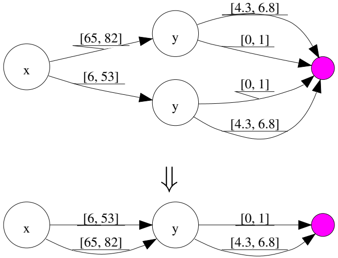


Isomorphic nodes

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. $\forall E \in \mathrm{Out}(N_1), \mathrm{Itv}(E) = \mathrm{Itv}(\sigma(E))$ and $\mathrm{Dest}(E) = \mathrm{Dest}(\sigma(E))$.

Reduction of an IA: merging of isomorphic nodes

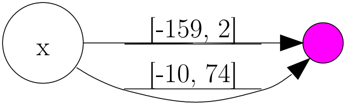
We described several operations allowing to reduce an IA's size without changing its semantics.



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. $\forall E \in \mathrm{Out}(N_1), \mathrm{Itv}(E) = \mathrm{Itv}(\sigma(E))$ and $\mathrm{Dest}(E) = \mathrm{Dest}(\sigma(E))$.

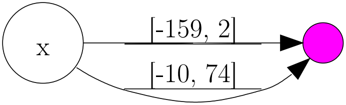
Reduction of an IA: merging of contiguous edges



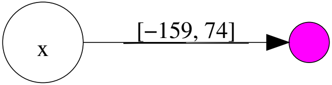
Contiguous edges

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 $I \in \mathbb{R}$ s.t. $\mathrm{Itv}(E_1) \cup \mathrm{Itv}(E_2) = I \cap \mathrm{Dom}(\mathrm{Var}(E_1))$.

Reduction of an IA: merging of contiguous edges



$\Downarrow$

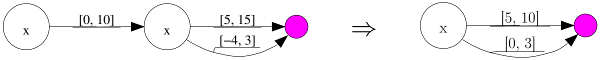


Contiguous edges

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 $I \in \mathbb{R}$ s.t. $\mathrm{Itv}(E_1) \cup \mathrm{Itv}(E_2) = I \cap \mathrm{Dom}(\mathrm{Var}(E_1))$.

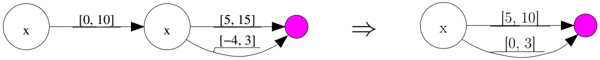
Reduction of an IA

Merging of stammering nodes:

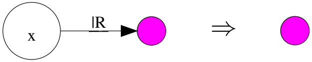


Reduction of an IA

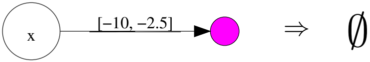
Merging of stammering nodes:



Elimination of undecisive nodes:

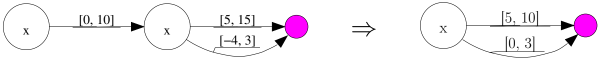


Elimination of unreachable edges (here $\mathrm{Dom}(x) = \mathbb{R}_+$):

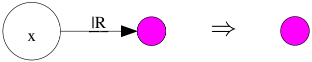


Reduction of an IA

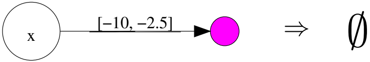
Merging of stammering nodes:



Elimination of undecisive nodes:



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

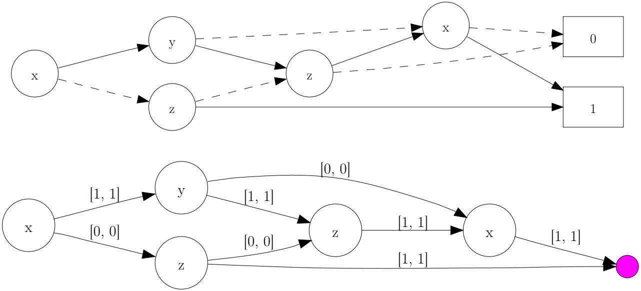


Proposition: reduction of an IA

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

Relation with BDD

IAs are a generalization of BDDs to continuous variables: BDDs are particular IAs (Boolean variables, deterministic nodes).



As in a BDD, a variable can be repeated on a path, and the order is not important.

Outline

Using a policy

Two operations needed online:

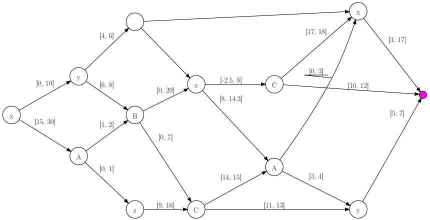
- conditioning (assign state variables according to the observations);
- model extraction (produce one decision among the possible ones).

Using a policy

Two operations needed online:

- conditioning (assign state variables according to the observations);
- model extraction (produce one decision among the possible ones).

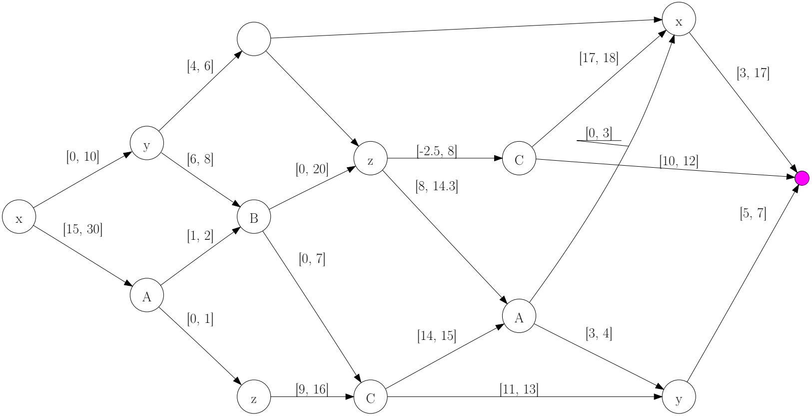
Illustration on a simple example, with three state variables x, y, z and three decision variables A, B, C:



Conditioning

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

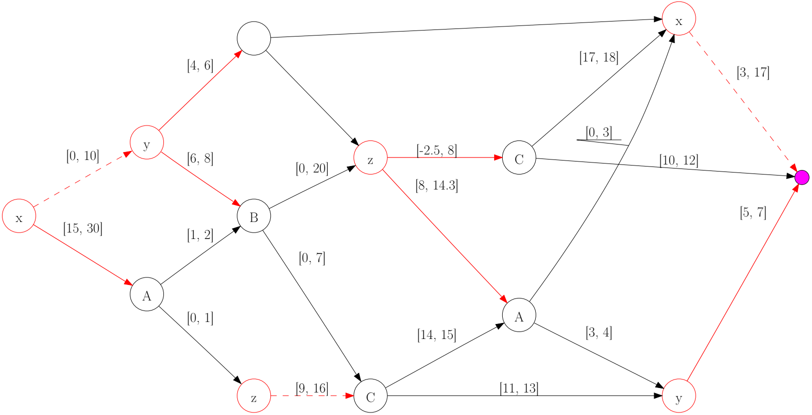
$\longrightarrow$ Conditioning of the policy :



Conditioning

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

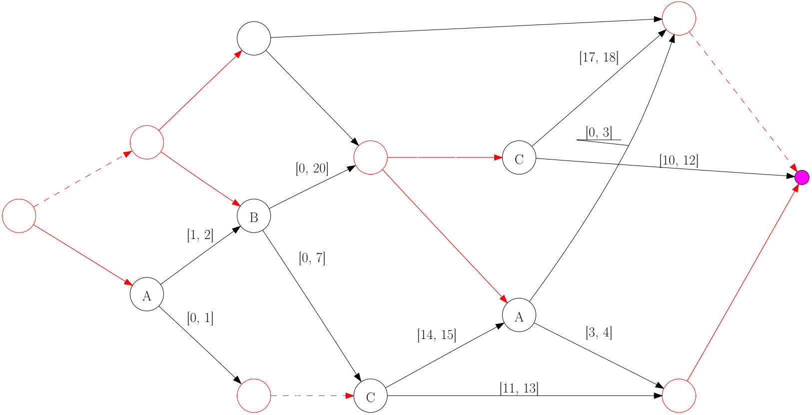
$\longrightarrow$ Conditioning of the policy :



Conditioning

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

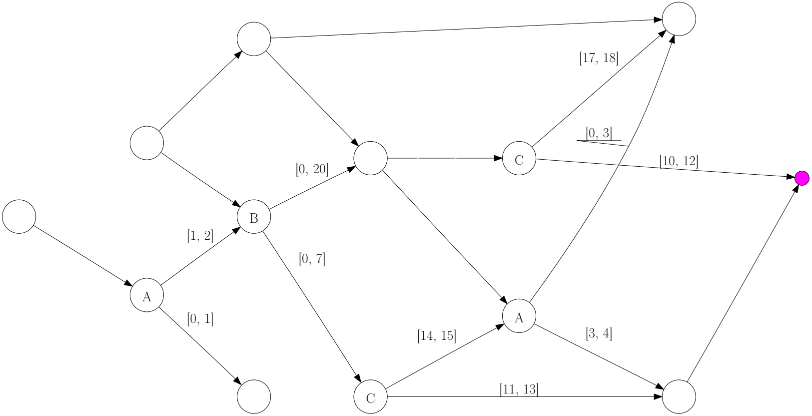
$\longrightarrow$ Conditioning of the policy :



Conditioning

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

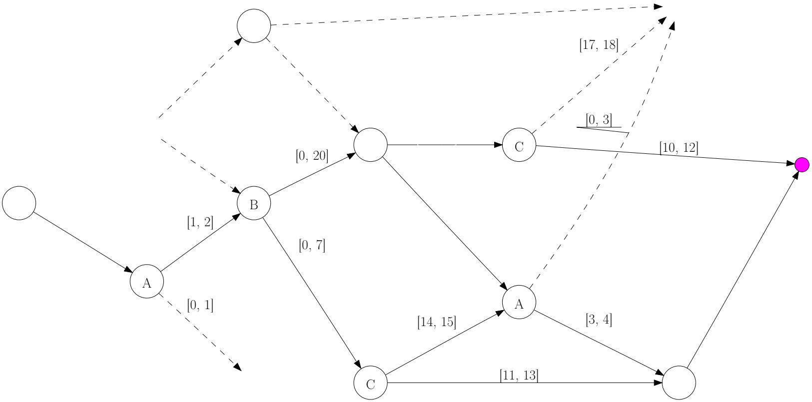
$\longrightarrow$ Conditioning of the policy :



Conditioning

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

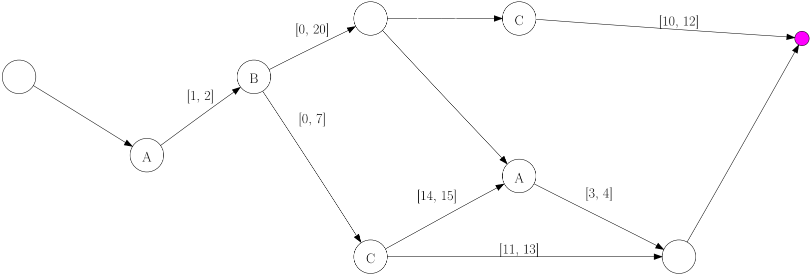
$\longrightarrow$ Conditioning of the policy :



Conditioning

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

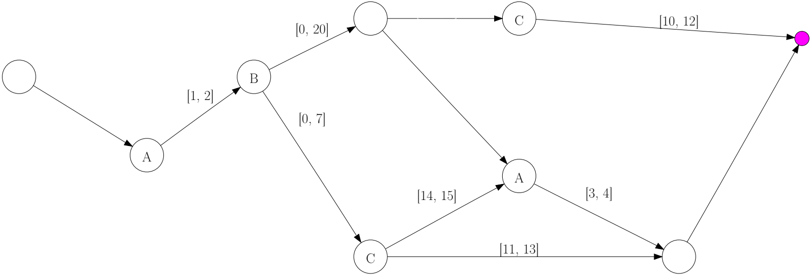
$\longrightarrow$ Conditioning of the policy :



Model extraction

We obtain a set of suitable decisions.

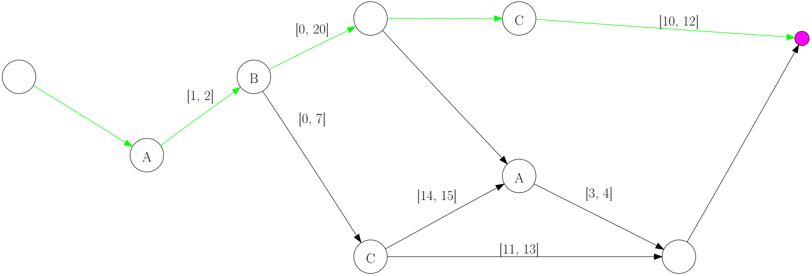
We need to chose one of them:



Model extraction

We obtain a set of suitable decisions.

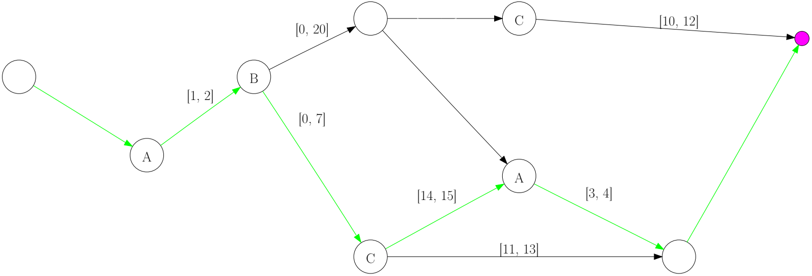
We need to chose one of them:



Model extraction

We obtain a set of suitable decisions.

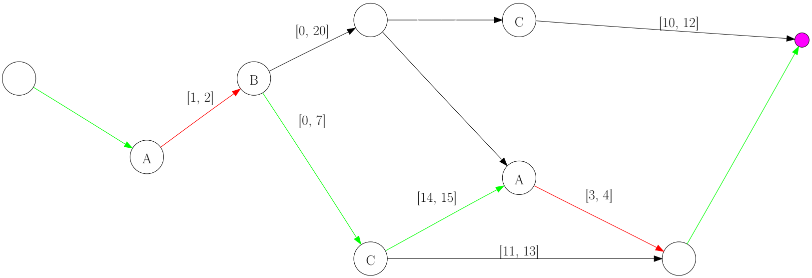
We need to chose one of them:



Model extraction

We obtain a set of suitable decisions.

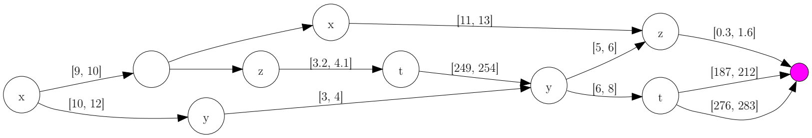
We need to chose one of them:



Model extraction

Problem: paths in an IA (even reduced) can be inconsistent.

A reduced IA can even have no consistent path at all:



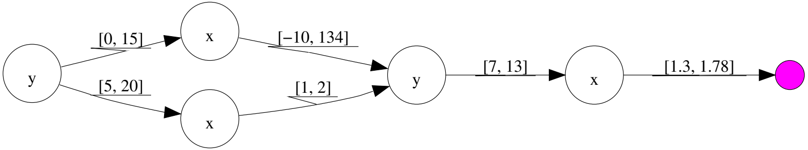
Same as for BDDs, this makes model extraction hard.

$\longrightarrow$ restriction on IAs, to make this operation easier.

Outline

Focusing interval automata (FIAs)

Idea: force intervals to shrink along a path; this property is called *focusingness*.



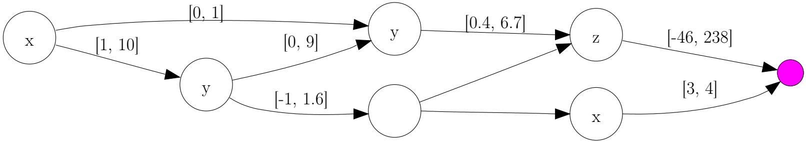
Focusing interval automaton

An IA is *focusing* iff for each edge E, all edges E' on a path from the root to $\mathrm{Src}(E)$ such that $\mathrm{Var}(E) = \mathrm{Var}(E')$ verify $\mathrm{Itv}(E) \subseteq \mathrm{Itv}(E')$.

FIAs (focusing IAs) allow the two desired operations (conditioning and model extraction) in polytime.

Relation with FBDD

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



This restriction makes model extraction easy, in the same manner as the “read-once” property on BDDs (which gives FBDDs).

FBDDs (and OBDDs) are particular FIAs.

Operations supported by IAs and FIAs

Table of supported operations:

Queries	IA	FIA	DNNF
CO(consistency)	$\circ$	$\surd$	$\surd$
VA(validity)	$\circ$	$\circ$	$\circ$
EQ(équivalence)	$\circ$	$\circ$	$\circ$
MC(model checking)	$\surd$	$\surd$	$\surd$
MX(model extr.)	$\circ$	$\surd$	$\surd$
CX(context extr.)	$\circ$	$\surd$	$\surd$
ME(model enum.)	$\circ$	$\surd$	$\surd$

Transformations	IA	FIA	DNNF
CD(conditioning)	$\surd$	$\surd$	$\surd$
$\wedge$**tC**(conj. with a term)	$\surd$	$\surd$	$\surd$
FO(forgetting)	$\circ$	$\surd$	$\surd$
SFO(simple forg.)	$\surd$	$\surd$	$\surd$
EN(ensuring)	$\circ$	$\circ$	$\circ$
SEN(simple ens.)	$\surd$	$\circ$	$\circ$
$\wedge$**C**(conjunction)	$\surd$	$\circ$	$\circ$
$\wedge$**BC**(binary conj.)	$\surd$	$\circ$	$\circ$
$\vee$**C**(disjunction)	$\surd$	$\surd$	$\surd$

$\surd$ means “supports in polytime”, $\circ$ “does not support, except if $P = NP$”.

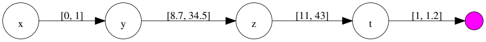
Outline

Building FIAs by "pushing boxes"

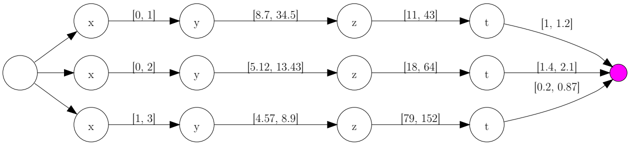
Using an algorithm provides a list of admissible boxes:

- It is straightforward to build the FIA representing a "box"

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



- Disjunction ($\vee$**C**) is easy on FIAs

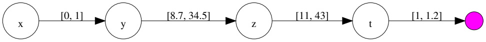


Building FIAs by "pushing boxes"

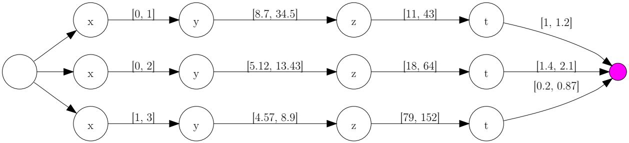
Using an algorithm provides a list of admissible boxes:

- It is straightforward to build the FIA representing a "box"

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



- Disjunction ($\vee$**C**) is easy on FIAs



Building FIAs by the trace of a solver

Using the trace of an interval-based CSP solver, following the approach of [Huang and Darwiche, 2005]:

Principle

The solver slices the domains until it finds a box either entirely contained in the model set, or of size lower than a given threshold.
Our algorithm then creates new nodes, and merges them with the current FIA.

We applied this method with the RealPaver solver [Granvilliers and Benhamou, 2006].

Outline

Problems

`obsmem` : manages connections between the observation device and the mass memory of a satellite.

`robot` : a robot exploring an area.

`ring` : standard benchmark for planning with non-determinism.

`drone` : a drone must achieve different goals on a number of zones in limited time. (two versions : discrete and hybrid)

Results

problem	red. time (ms)	size (edges)	% input	% OBDD	**CD**(ms)	**MX**(ms)
obsmem2	1102	100	74	66	1	5
obsmem3	2168	197	75	69	4	11
obsmem4	4729	342	75	70	4	11
obsmem5	5657	546	76	70	7	19
obsmem6	9433	820	76	76	11	35
porobot	4035	56	97	36	0	1
forobot	52767	60	99	31	0	3
ring7	92	13	75	71	0	1
ring8	185	13	78	75	0	1
ring9	92	13	80	75	0	1
ring10	82	13	81	75	0	2
drone10	46732	453	95	47	11	23
drone20	947174	763	97	44	30	61
drone30	2850715	944	98	43	21	48
drone40	5721059	944	98	45	15	29
drone10	104373	16820	35	×	7143	110
drone20	418885	38076	35	×	16970	193
drone30	1850326	53917	36	×	23597	612

Conclusion

We decribed a compilation structure theoretically allowing to represent and exploit online decision policies involving both discrete and continuous variables.

Significant gain in size in comparison to OBDDs.

Operations duration is interesting, yet to be improved.

Future work: Improvement of algorithms, and comparison with other non-Boolean structures (finite-state automata [Vempaty, 1992]).

Perspectives

Depending on the results:

- Search for better building methods for FIAs (heuristics. . .)
- Integration of valuations in IAs, approximate compilation [Venturini and Provan, 2008]
- Other structures (R^*-*trees*. . .)

References

Granvilliers, L. and Benhamou, F. (2006).
Algorithm 852: RealPaver: an Interval Solver Using Constraint Satisfaction Techniques.
ACM Trans. Math. Softw., 32(1):138–156.

Huang, J. and Darwiche, A. (2005).
DPLL with a Trace: From SAT to Knowledge Compilation.
In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 156–162.

Vempaty, N. R. (1992).
Solving Constraint Satisfaction Problems Using Finite State Automata.
In *Association for the Avancement of Artificial Intelligence Conference (AAAI)*, pages 453–458.

Venturini, A. and Provan, G. (2008).
Incremental Algorithms for Approximate Compilation.
In *Association for the Avancement of Artificial Intelligence Conference (AAAI)*, pages 1495–1499.