EVMDD Algorithms for Distance Functions

Pierre Roux[1] **Radu I. Siminiceanu**[2]

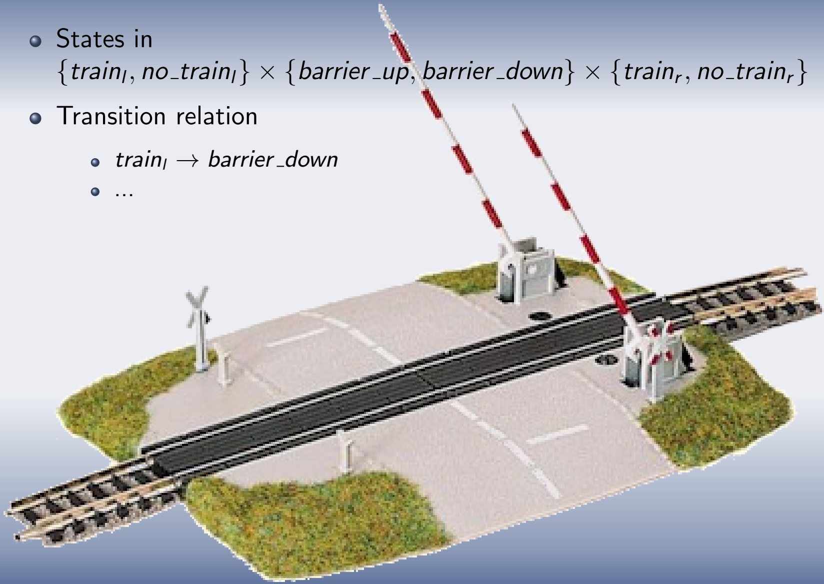
August 19, 2010

[1] École Normale Supérieure de Lyon, France (pierre.roux@ens-lyon.fr)

[2] NIA (radu@nianet.org)

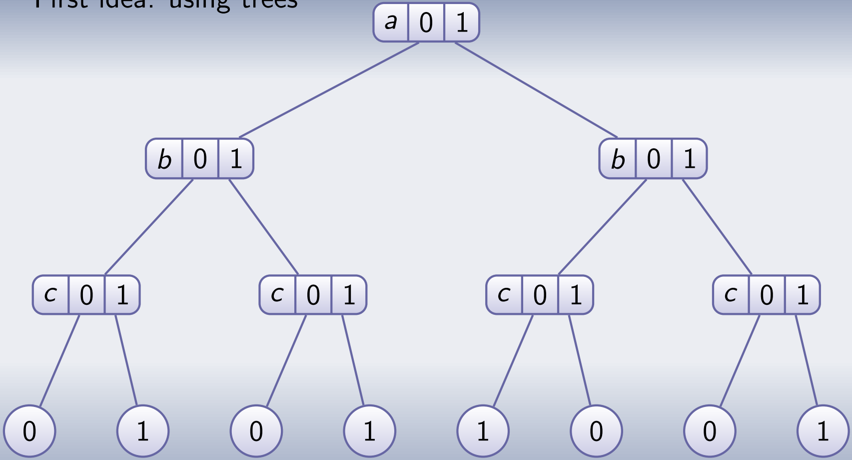
Model Checking

- States in $\{train_l, no\_train_l\} \times \{barrier\_up, barrier\_down\} \times \{train_r, no\_train_r\}$
- Transition relation
  - $train_l \rightarrow barrier\_down$
  - ...



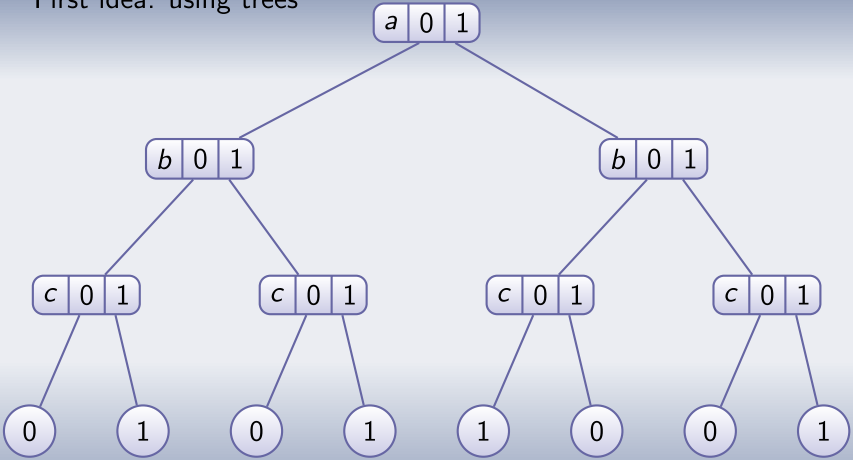
Representing boolean functions

First idea: using trees



Representing boolean functions

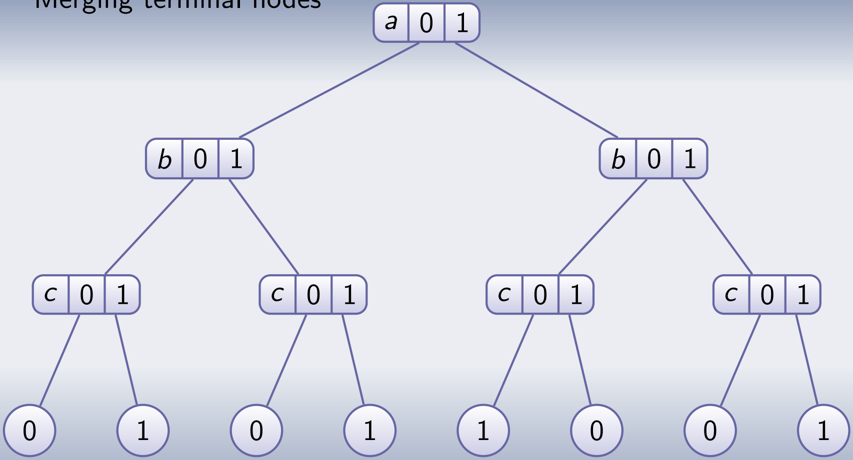
First idea: using trees



requires 2^n terminal nodes

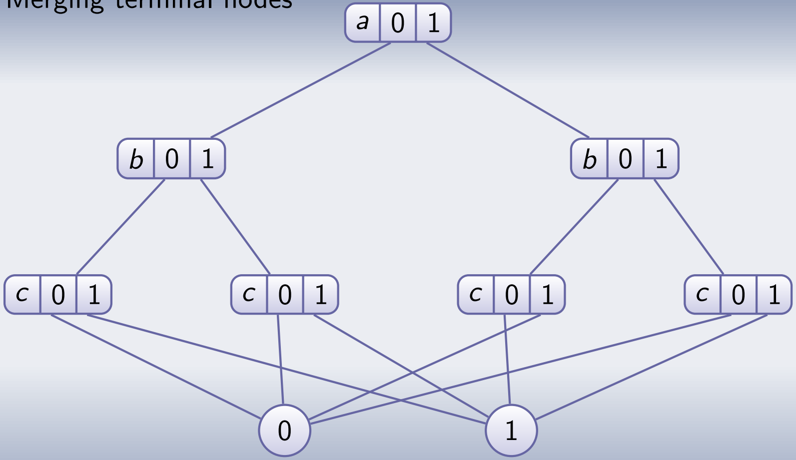
Binary Decision Diagram (BDD)

Merging terminal nodes



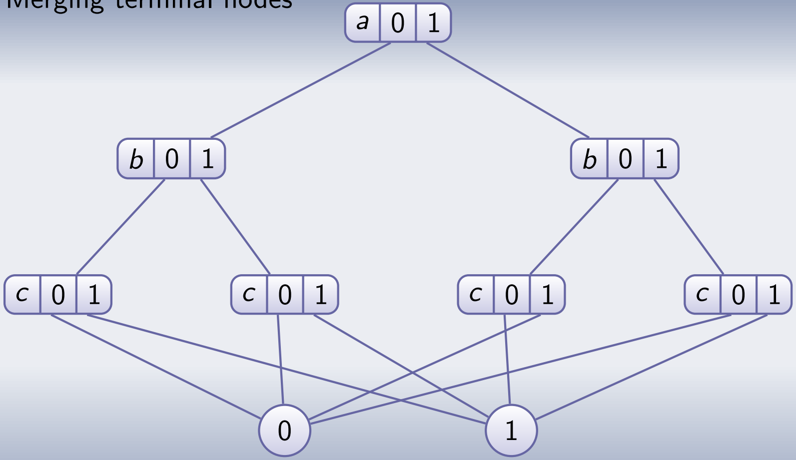
Binary Decision Diagram (BDD)

Merging terminal nodes



Binary Decision Diagram (BDD)

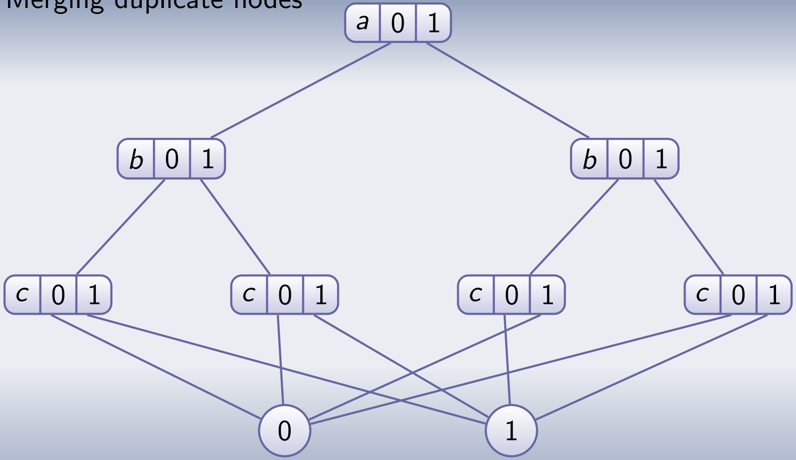
Merging terminal nodes



still exponential

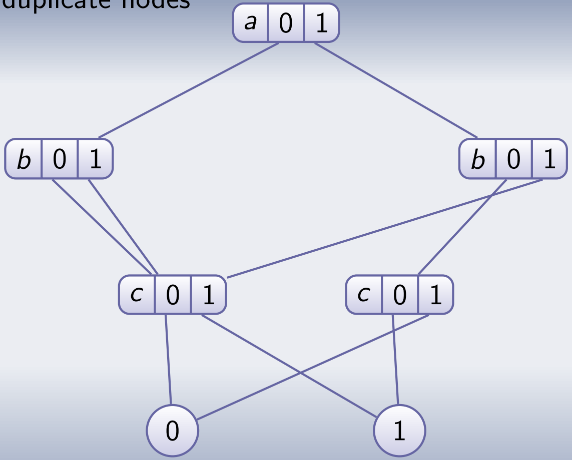
BDD cont'd

Merging duplicate nodes



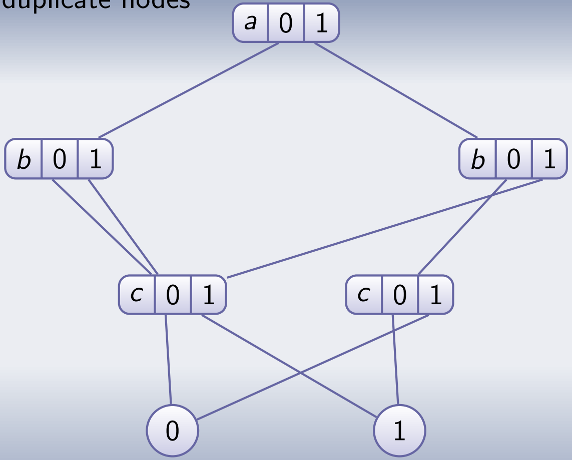
BDD cont'd

Merging duplicate nodes



BDD cont'd

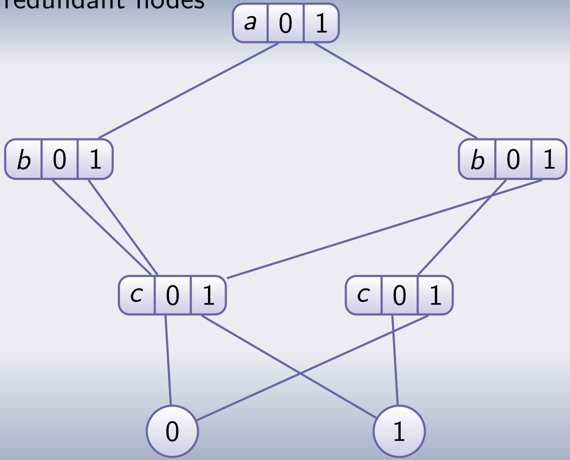
Merging duplicate nodes



exponential in worst case, often better in practice

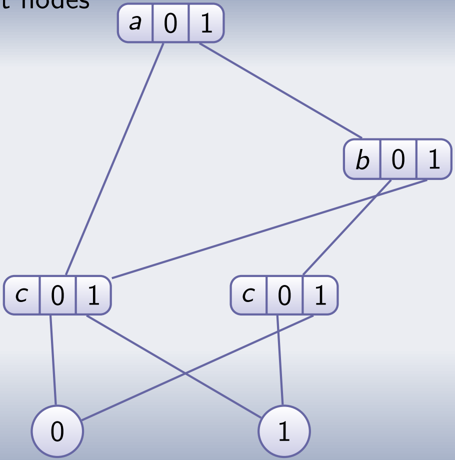
BDD end

Deleting redundant nodes



BDD end

Deleting redundant nodes



BDD characteristics

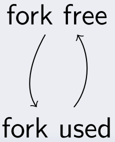
- canonicity:
  two BDD represent the same function iff they are isomorphic
- easy computation:
  for f, g represented by BDDs of size $|f|$ and $|g|$
  $f * g$ computed in $\mathrm{O}(|f|\,|g|)$
  using dynamic programming techniques
- size of a BDD completely **unrelated** to size of represented set

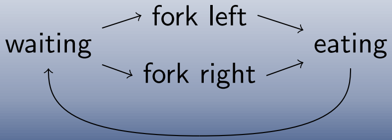
Globally asynchronous, locally synchronous models

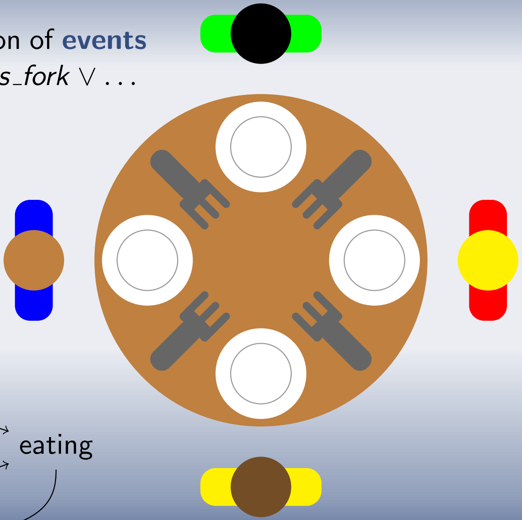
transition relation: disjunction of **events**

$phil_0\_takes\_fork \vee phil_1\_takes\_fork \vee \ldots$

e.g. **Dining Philosophers**







Breadth First Search

BFS($initial\_state$: BDD, $events$: BDD list) : BDD
  $r \leftarrow \emptyset$
  $frontier \leftarrow initial\_state$
  do
    $oldr \leftarrow r$
    $next \leftarrow \emptyset$
    for all e **in** $events$ **do**
      $next \leftarrow next \cup e(frontier)$
    done
    $frontier \leftarrow next \setminus r$
    $r \leftarrow r \cup frontier$
  while $r \neq oldr$
  return r

Alternative BFS

BFS($initial\_state$: BDD, $events$: BDD list) : BDD

$r \leftarrow initial\_states$
do
$oldr \leftarrow r$
$next \leftarrow \emptyset$
for all e **in** $events$ **do**
$next \leftarrow next \cup e(r)$
done
$r \leftarrow r \cup next$
while $r \neq oldr$
return r

Works often **better**
(size of BDD r unrelated to number of encoded states)

BFS with chaining and saturation

BFS(*initial_state* : BDD, *events* : BDD list) : BDD
  $r \leftarrow \emptyset$
  do
    $oldr \leftarrow r$
    for all e **in** *events* **do**
      $r \leftarrow r \cup e(r)$
    done
  while $r \neq oldr$
  return r

Often a **lot better**

ultimately:
Saturation: compute **local fixpoints** instead of one global fixpoint

Saturation vs BFS

Model	Model size	Reachable states	BFS (sec)	Saturation (sec)
Dining philosophers	100	5×10^{62}	0.66	0.01
	1000	9×10^{626}	145.06	0.04
	10000	4×10^{6269}	—	0.43
Round robin mutual exclusion protocol	50	1×10^{17}	4.63	0.12
	100	3×10^{32}	104.24	1.04
	200	7×10^{62}	—	8.36
Slotted ring protocol	20	3×10^{20}	5.79	0.03
	40	4×10^{41}	258.55	0.22
	80	1×10^{84}	—	1.74

On Intel Core 2, 1.2GHz, 1.5GB mem
("—" means "out of memory").

Saturation vs BFS

Model	Model size	Reachable states	BFS (sec)	Saturation (sec)
Kanban assembly line	40	1×10^{15}	11.11	0.06
	200	3×10^{22}	—	8.54
	400	6×10^{25}	—	90.14
Knights problem	5	7×10^{7}	345.25	0.22
	7	2×10^{15}	—	2.05
	9	9×10^{24}	—	9.22
Randomized leader election protocol	6	2×10^{6}	4.48	0.87
	8	4×10^{8}	78.76	7.48
	9	5×10^{9}	258.69	18.98

On Intel Core 2, 1.2GHz, 1.5GB mem
("—" means "out of memory").

Distance function

- Distance to initial state, useful for:
  - knowing the **diameter** of the model (greatest distance)
  - getting a **shortest counterexample** to a property in temporal logic

Distance function

- Distance to initial state, useful for:
  - knowing the **diameter** of the model (greatest distance)
  - getting a **shortest counterexample** to a property in temporal logic
- Implicitly computed by BFS:
  at nth step, $r = \{s \mid s \textit{ at distance at most } n\}$

Distance function

- Distance to initial state, useful for:
  - knowing the **diameter** of the model (greatest distance)
  - getting a **shortest counterexample** to a property in temporal logic
- Implicitly computed by BFS:
  at nth step, $r = \{s \mid s \textit{ at distance at most } n\}$
- But neither by BFS with chaining nor by saturation
- A modified version of saturation keeps track of distances using another data structure: edge valued decision diagrams

Multiple Terminal BDD (MTBDD)

- $f : \{0,1\}^n \rightarrow \mathbb{Z}$

Multiple Terminal BDD (MTBDD)

- $f : \{0,1\}^n \to \mathbb{Z}$
- Extend BDD to Multiple Terminal BDD

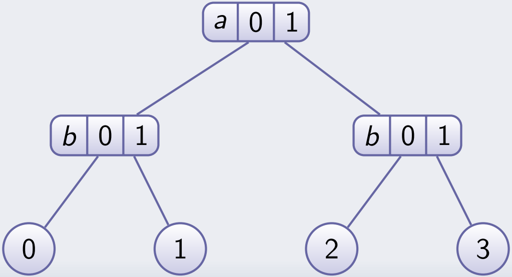


Figure: $f : (a, b) \mapsto 2a + b$

Multiple Terminal BDD (MTBDD)

- $f : \{0,1\}^n \to \mathbb{Z}$
- Extend BDD to Multiple Terminal BDD

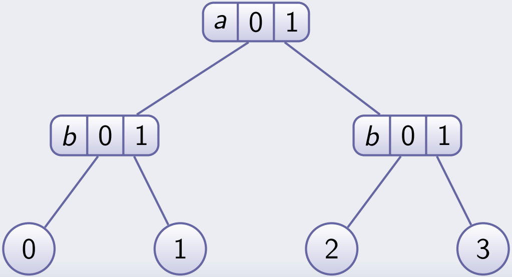
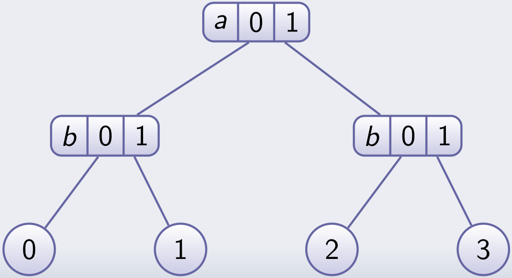


Figure: $f : (a, b) \mapsto 2a + b$

- Bad if $\mathrm{Img}(f)$ too big

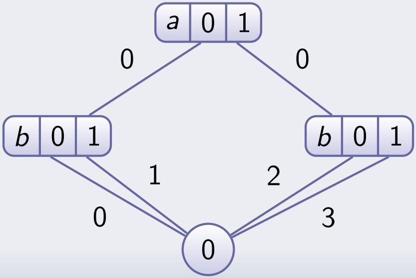
Edge Valued DD (EVBDD)

Merging all terminals to 0 and putting values on edges



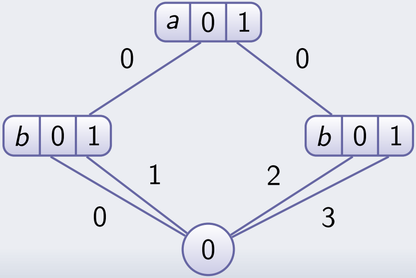
Edge Valued DD (EVBDD)

Merging all terminals to 0 and putting values on edges



Edge Valued DD (EVBDD)

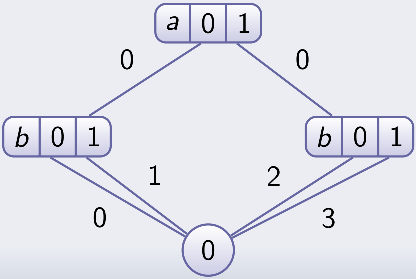
Merging all terminals to 0 and putting values on edges



Result: sum of edge values on path from root to terminal node

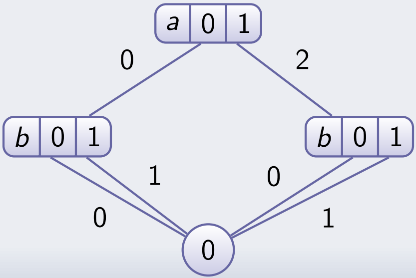
EVBDD cont'd

Canonical node: minimum of outgoing edge values is 0



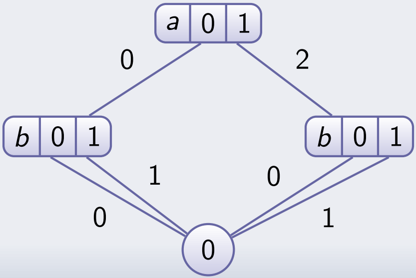
EVBDD cont'd

Canonical node: minimum of outgoing edge values is 0



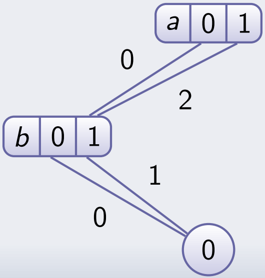
EVBDD end

Merging duplicate nodes



EVBDD end

Merging duplicate nodes



Using reachability information

- Saturation is faster than any distance computation
- We can then
  - first compute reachable state space
  - and use it to improve distance computation
- simplest idea: don't try to reach a state already known unreachable

Comparison of algorithms

Model	Model size	BFS (sec)	Saturation (sec)	Distance (sec)	Modified (sec)
Dining philosophers	100	0.66	0.00	0.00	0.00
	1000	145.06	0.04	0.04	0.05
	10000	—	0.43	0.41	0.52
Round robin mutual exclusion protocol	50	4.63	0.12	0.13	0.02
	100	104.24	1.04	1.08	0.08
	200	—	8.36	8.30	0.37
Slotted ring protocol	20	5.79	0.03	0.13	0.13
	40	258.55	0.22	1.50	1.59
	80	—	1.74	20.54	21.70

On Intel Core 2, 1.2GHz, 1.5GB mem
("—" means "out of memory").

Comparison of algorithms

Model	Model size	BFS (sec)	Saturation (sec)	Distance (sec)	Modified (sec)
Kanban assembly line	40	11.11	0.06	0.08	0.10
	200	—	8.54	10.50	10.86
	400	—	90.14	109.88	104.46
Knights problem	5	345.25	0.22	—	—
	7	—	2.05	—	—
	9	—	9.22	—	—
Randomized leader election protocol	6	4.48	0.87	2.26	2.77
	8	78.76	7.48	28.55	64.54
	9	258.69	18.98	168.43	284.86

On Intel Core 2, 1.2GHz, 1.5GB mem
("—" means "out of memory").

Comparison of algorithms

Model	Model size	BFS (sec)	Saturation (sec)	Distance (sec)	Modified (sec)
Virtual Filesystem	3	0.09	0.10	0.14	0.14
	4	0.75	1.00	1.34	1.30
	5	5.04	6.35	8.29	7.84
Runway Safety Monitor	3, 5, 3	0.07	0.82	1.71	0.92
	4, 6, 4	0.17	6.53	12.63	12.68
	5, 7, 5	0.28	26.95	66.08	53.16
Rubik's cube	2	85.78	1.92	—	—
	3	—	—[1]	—	—

On Intel Core 2, 1.2GHz, 1.5GB mem
("—" means "out of memory").

[1]half an hour on another machine with 40GB of RAM

Bounded saturation

- sometimes interested only in “close” states
- with BFS: stop after *bound* iterations
- with saturation:
  - simplest solution: after firing an event,
    forget states which are further than *bound*
  - or: start from initial state at distance 0 and state space at *bound*
    and don’t try to fire events which can not make any state closer

Bounded saturation (poor) results

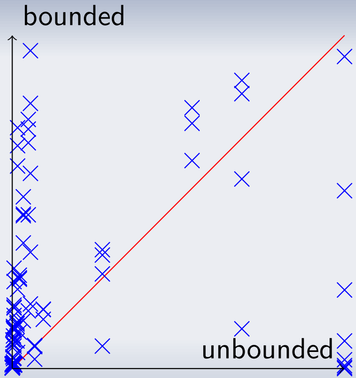


Figure: Bounded vs unbounded saturation

Bounded saturation (poor) results, cont'd

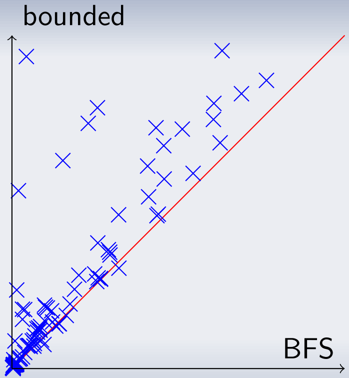


Figure: Bounded saturation vs BFS

Bounded saturation (poor) results, cont'd

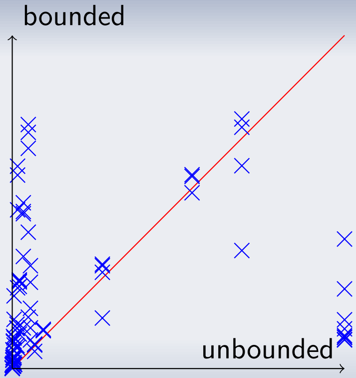


Figure: Bounded from state space vs unbounded saturation

Bounded saturation (poor) results, end

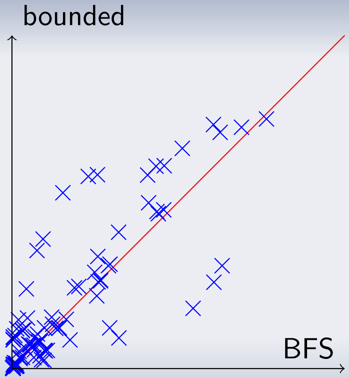


Figure: Bounded from state space saturation vs BFS

Conclusions

- considering events affecting same variables as one big event is often a good idea, but not always
- if those events are not merged, order matters

Conclusions

- considering events affecting same variables as one big event is often a good idea, but not always
- if those events are not merged, order matters
- saturation with EVBDD is the best to compute distances, when it works
- otherwise, BFS remains the best

Conclusions

- considering events affecting same variables as one big event is often a good idea, but not always
- if those events are not merged, order matters
- saturation with EVBDD is the best to compute distances, when it works
- otherwise, BFS remains the best
- models on which distance saturation doesn't work are those with moves in 2D/3D spaces:
  - knights on chess board (knights problem)
  - planes in sky (runway safety monitor)
  - ...
- this prevents a good locality of events

Future directions

- could reachability information be used to improve BFS ?
- 2D/3D efficient encoding
  - basic idea: BDD with children in two (or three) directions
  - appealing but prevents coding of any relation between directions
  - other ideas to be explored:
    - unordered BDDs
    - hierarchical set decision diagrams

Questions

?