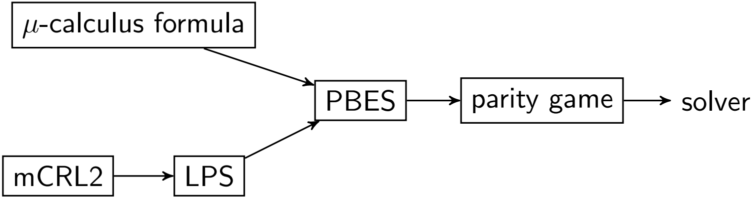
Solving infinite parity games through bisimulation quotienting

Dutch Model Checking Day

Thomas Neele
Joint work with: Tim Willemse and Jan Friso Groote

June 21st, 2018

Model checking with mCRL2



Parity Game

A parity game:

- has two players: $\Diamond$ (even) and $\Box$ (odd), who move a token;
- is played on a directed graph;
- every node has a priority.

Parity Game

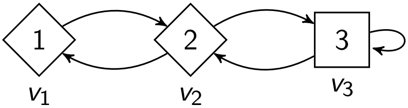
A parity game:

- has two players: $\Diamond$ (even) and $\Box$ (odd), who move a token;
- is played on a directed graph;
- every node has a priority.

Definition

A *parity game* is a graph $(V, \to, \Omega, \mathcal{P})$, where:

- V is a set of nodes;
- $\to \subseteq V \times V$ is a left-total transition relation;
- $\Omega : V \to \mathbb{N}$ is a function that assigns a priority to nodes;
- $\mathcal{P} : V \to \{\Diamond, \Box\}$ is a function that assigns players to nodes.



Winning paths

Definition

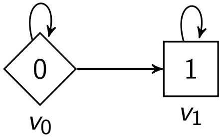
An infinite path π, called a *play*, is winning for a player $\Diamond$ if the minimal priority that occurs infinitely often on π is even. Otherwise, it is winning for $\Box$.

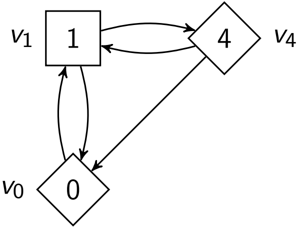
Winning paths

Definition

An infinite path π, called a *play*, is winning for a player $\Diamond$ if the minimal priority that occurs infinitely often on π is even. Otherwise, it is winning for $\Box$.

Find winning plays:





Strategies

Definition

A strategy $S_p : V \rightarrow V$ for a player p is a partial-function that is only defined on the vertices of p.

Strategies

Definition

A strategy $S_p : V \to V$ for a player p is a partial-function that is only defined on the vertices of p.

Definition

A play $\pi = v_0 v_1 \ldots$ is *consistent* with a strategy S iff for all v_i such that $S(v_i)$ is defined, $S(v_i) = v_{i+1}$.

Strategies

Definition

A strategy $S_p : V \to V$ for a player p is a partial-function that is only defined on the vertices of p.

Definition

A play $\pi = v_0 v_1 \dots$ is *consistent* with a strategy S iff for all v_i such that $S(v_i)$ is defined, $S(v_i) = v_{i+1}$.

Definition

A strategy S is a *winning strategy* in vertex v iff all plays starting in v that are consistent with S are winning.

Strategies

Definition

A strategy $S_p : V \to V$ for a player p is a partial-function that is only defined on the vertices of p.

Definition

A play $\pi = v_0 v_1 \ldots$ is *consistent* with a strategy S iff for all v_i such that $S(v_i)$ is defined, $S(v_i) = v_{i+1}$.

Definition

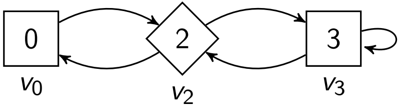
A strategy S is a *winning strategy* in vertex v iff all plays starting in v that are consistent with S are winning.

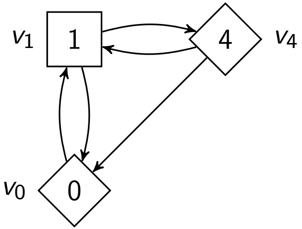
Definition

A vertex v is *won by a player* p iff there is a winning strategy for p in v.

Examples

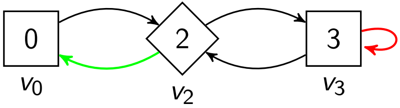
Find winning strategies:

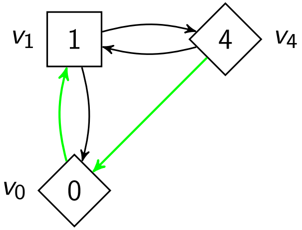




Examples

Find winning strategies:

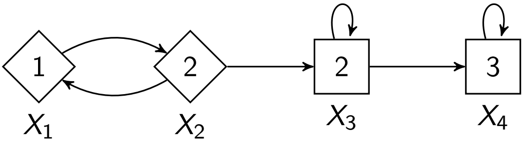




From parity game to BES

- A *Boolean equation system* is a sequence of fixpoint equations over Boolean variables.
- A BES in *standard recursive form* can be mapped directly to a parity game:
    - One node for every equation;
    - One outgoing transition for every variable in the right-hand side of an equation;
    - Priority is determined by the fixpoint;
    - Player is determined by the operand.

$$\begin{aligned}\mu X_1 &= X_2\\ \nu X_2 &= X_1 \vee X_3\\ \nu X_3 &= X_3 \wedge X_4\\ \mu X_4 &= X_4\end{aligned}$$



Generalizing to PBES

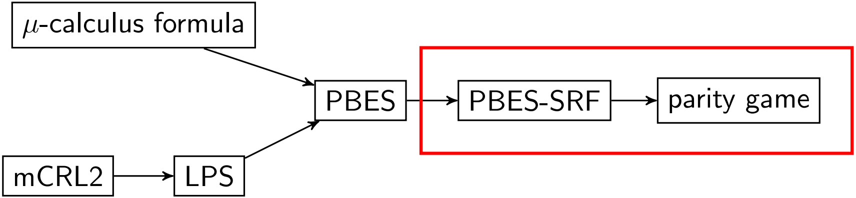
Parameterised Boolean equation system: a BES with data

$$\nu X(b{:}\mathbb{B}) = X(\neg b) \quad \xrightarrow{\mathit{instantiation}} \quad \begin{aligned} \nu X_{true} &= X_{false} \\ \nu X_{false} &= X_{true} \end{aligned}$$

PBESs can also contain expressions over data:

$$\mu Y(n{:}\mathbb{N}) = n \leq 2 \wedge Y(n+1) \quad \xrightarrow{\mathit{instantiation}} \quad \begin{aligned} \nu Y_0 &= Y_1 \\ \nu Y_1 &= Y_2 \\ \nu Y_2 &= Y_3 \\ \nu Y_3 &= \mathit{false} \\ \nu Y_4 &= \mathit{false} \\ &\vdots \end{aligned}$$

The complete picture



Bisimulation for parity games

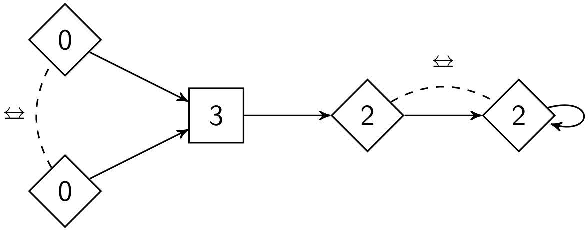
Definition

Let $G = (V, \rightarrow, \Omega, \mathcal{P})$ be a parity game. Then, a relation $R \subseteq V \times V$ is a bisimulation relation if for all $s \, R \, t$:

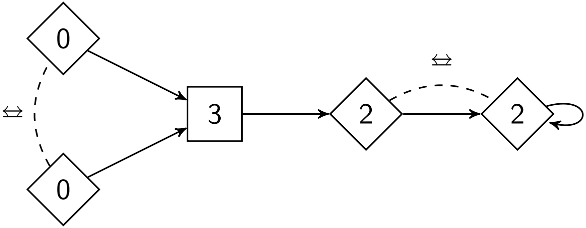
- $\Omega(s) = \Omega(t)$ and $\mathcal{P}(s) = \mathcal{P}(t)$;
- for all $s \rightarrow s'$ there is a $t \rightarrow t'$ such that $s' \, R \, t'$; and
- for all $t \rightarrow t'$ there is a $s \rightarrow s'$ such that $s' \, R \, t'$.

Two nodes s and t are bisimilar ($s \leftrightarrow t$) iff there is an R such that $s \, R \, t$.

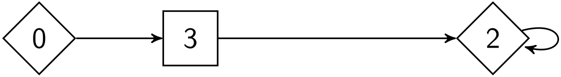
Bisimulation minimisation



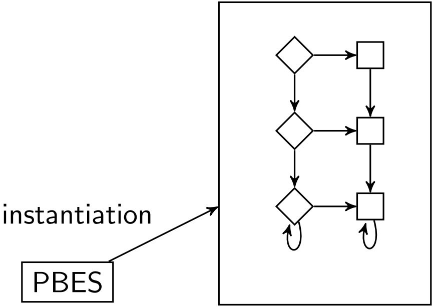
Bisimulation minimisation



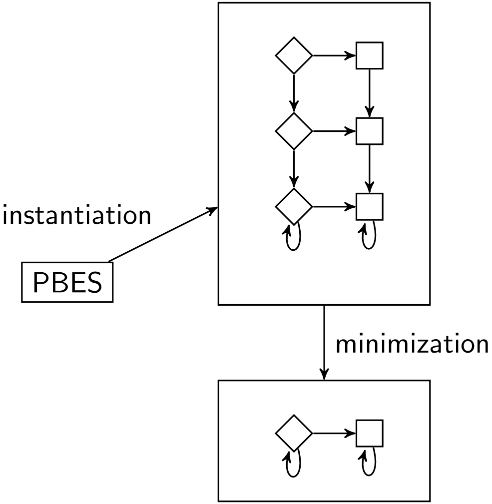
After merging bisimilar nodes:



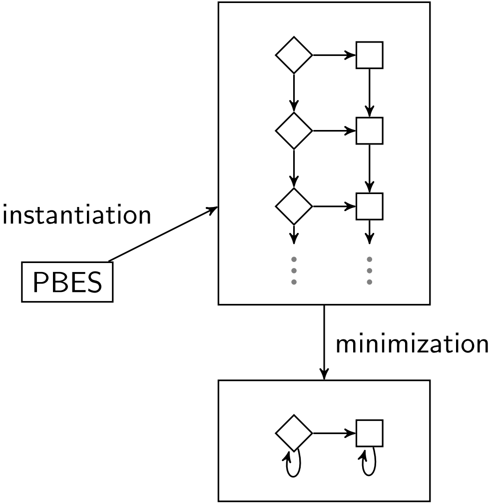
Minimal game generation



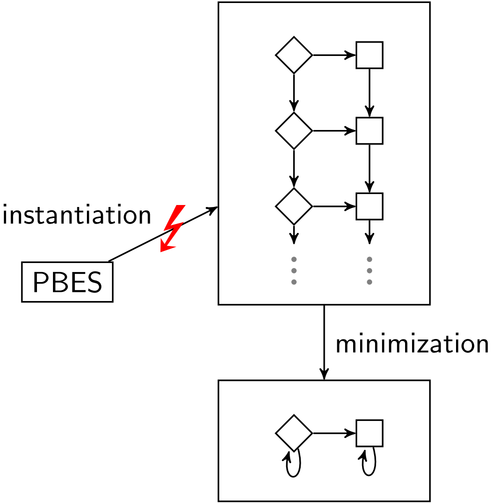
Minimal game generation



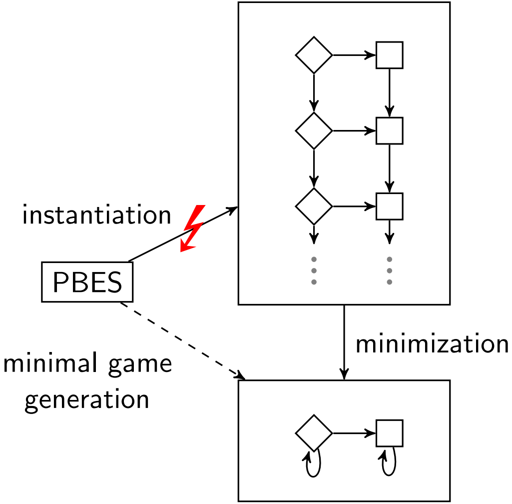
Minimal game generation



Minimal game generation



Minimal game generation



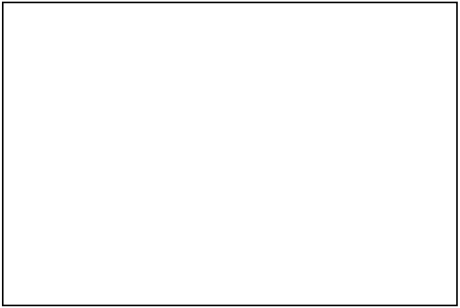
Some concepts

- block: set of nodes
- partition: set of pairwise disjunct blocks. The union over the blocks is equal to V.

Refinement algorithm

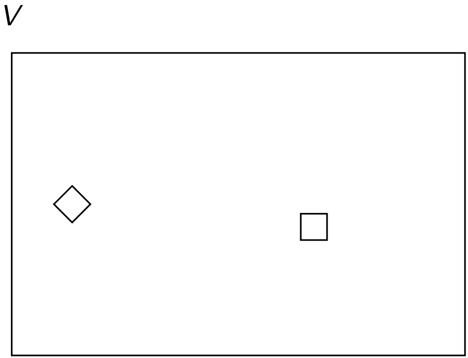
Every iteration: refine partition and remove unreachable blocks.

V



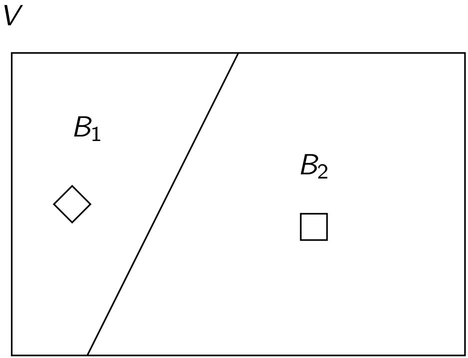
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



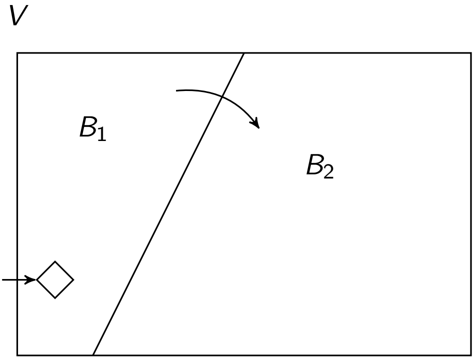
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



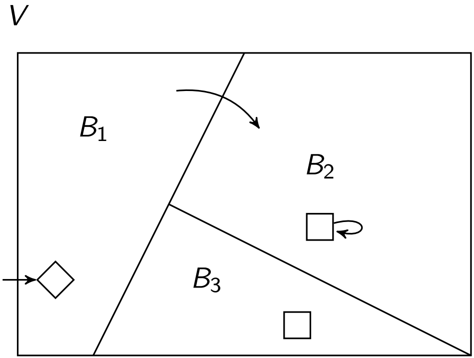
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



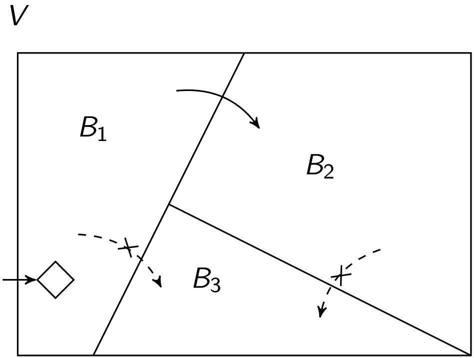
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



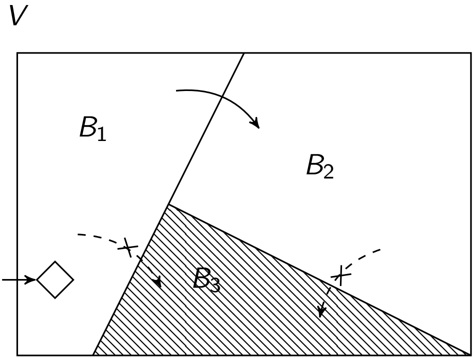
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



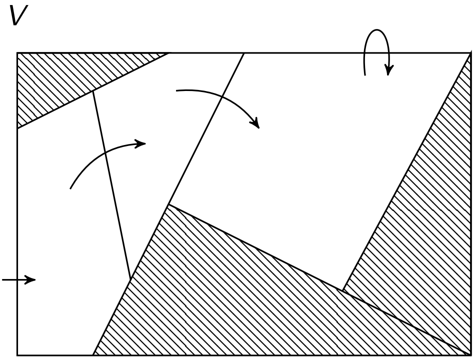
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



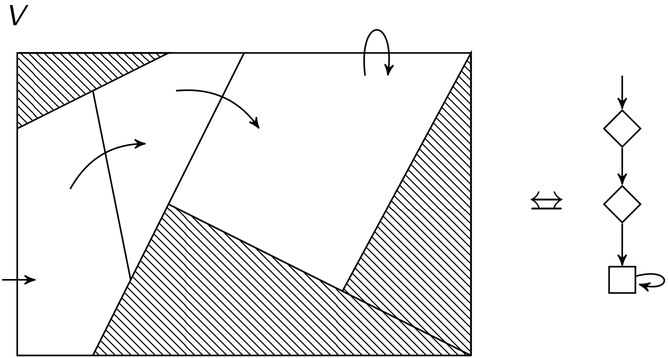
Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



Refinement algorithm

Every iteration: refine partition and remove unreachable blocks.



Limitations

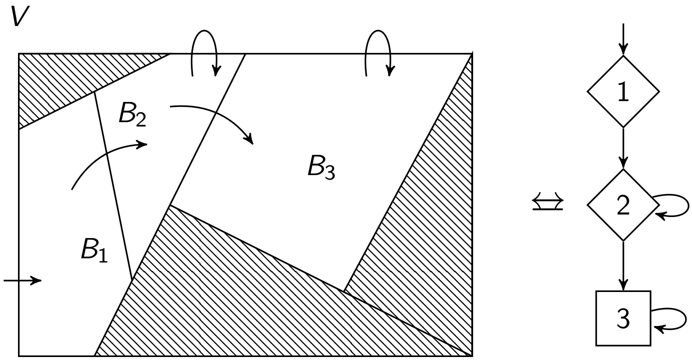
- Partition refinement does not terminate when the bisimulation quotient is infinite.
- Only a small part of the system might be relevant as witness/counter-example.

Solution: search for winning subgame, refine only those blocks.

Proof searching

Situation:

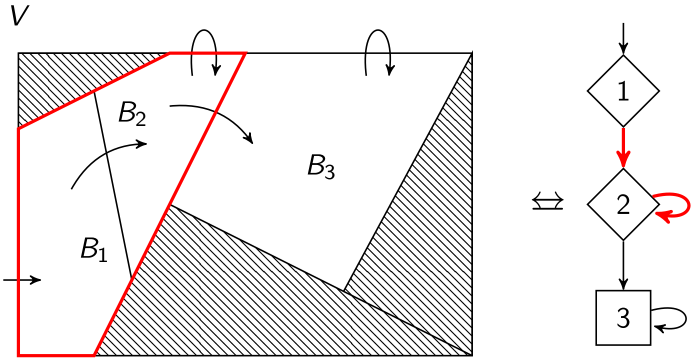
- Partition is not yet stable;
- The wining strategy for $\Diamond$ from the initial node does not involve B_3.



Proof searching

Situation:

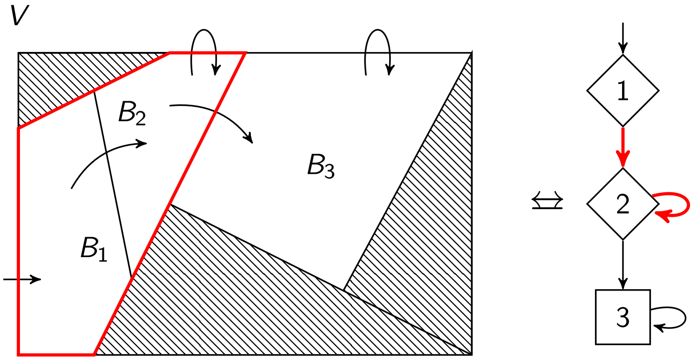
- Partition is not yet stable;
- The wining strategy for $\Diamond$ from the initial node does not involve B_3.



Proof searching

Theorem

If a winning subgame is stable wrt to itself, then this subgame is a valid witness/counter-example.



Conclusion

- Scalability increased through continuous searching for evidence.
- Approach is more generic than other symbolic approaches for PBESs.
- We can now model check all properties from the modal mu calculus on systems with infinite data domains.
- We can check equivalence of systems with infinite data through `lpsbisim2pbes`.

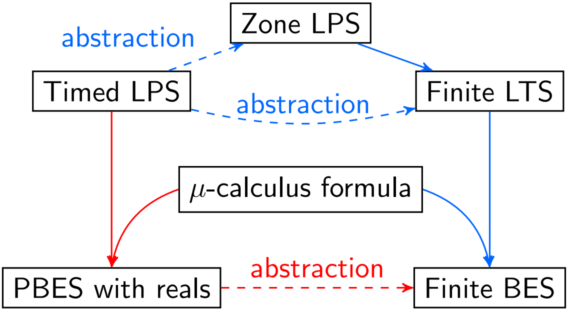
Future investigation:

- Find an optimal winning subgame.
- Weaker equivalence relation: idempotence-identifying bisimulation, simulation equivalence.

Thank you

Abstraction on different levels

- Sep 2016 to July 2017: on model level.
- This talk: on problem level.



Parameterised Boolean Equation System

Definition

A PBES is a sequence of equations as defined by the following grammar:

$$\mathcal{E} ::= \emptyset \mid (\nu X(d{:}D) = \phi)\mathcal{E} \mid (\mu X(d{:}D) = \phi)\mathcal{E}$$

where $\emptyset$ is the empty PBES and μ and ν denote the least and greatest fixpoint, respectively. Each *predicate variable* X is an element of some set of variables $\mathcal{X}$ and has type $D \rightarrow B$. Lastly, d is a parameter of type D.

Example:

$$\nu X(n{:}\mathbb{N}) = X(n+1) \vee Y(\mathit{true})$$
$$\mu Y(b{:}\mathbb{B}) = Y(\neg b)$$

Solution: $X(n)$ is true for all $n \in \mathbb{N}$, $Y(\mathit{true})$ and $Y(\mathit{false})$ are false.

Parallel worlds: generation/instantiation

$$\boxed{\text{LPS}} \longrightarrow \boxed{\text{LTS}}$$

LPS:

$$P(d{:}D) = \sum_{i \in I} \sum_{e_i : E_i} c_i(d, e_i) \rightarrow a_i(f_i(d, e_i)) \cdot P(g_i(d, e_i))$$

Parallel worlds: generation/instantiation

$$\boxed{\text{LPS}} \longrightarrow \boxed{\text{LTS}}$$

LPS:

$$P(d{:}D) = \sum_{i \in I} \sum_{e_i : E_i} c_i(d, e_i) \rightarrow a_i(f_i(d, e_i)) \cdot P(g_i(d, e_i))$$

$$\boxed{\text{PBES-SRF}} \longrightarrow \boxed{\text{parity game}}$$

disjunctive:

$$\sigma X(d{:}D) = \bigvee_{i \in I} \exists e_i{:}E_i.\ c_i(d, e_i) \wedge X_i(g_i(d, e_i))$$

conjunctive:

$$\sigma X(d{:}D) = \bigwedge_{i \in I} \forall e_i{:}E_i.\ c_i(d, e_i) \Rightarrow X_i(g_i(d, e_i))$$

Parallel worlds: generation/instantiation

$$\boxed{\text{LPS}} \longrightarrow \boxed{\text{LTS}}$$

LPS:

$$P(d{:}D) = \sum_{i \in I} \sum_{e_i : E_i} c_i(d, e_i) \rightarrow a_i(f_i(d, e_i)) \cdot P(g_i(d, e_i))$$

$$\boxed{\text{PBES-SRF}} \longrightarrow \boxed{\text{parity game}}$$

disjunctive:

$$\sigma X(d{:}D) = \bigvee_{i \in I} \exists e_i{:}E_i.\, c_i(d, e_i) \land X_i(g_i(d, e_i))$$

conjunctive:

$$\sigma X(d{:}D) = \bigwedge_{i \in I} \forall e_i{:}E_i.\, c_i(d, e_i) \Rightarrow X_i(g_i(d, e_i))$$

Parallel worlds: generation/instantiation

$$\boxed{\text{LPS}} \longrightarrow \boxed{\text{LTS}}$$

action-based

LPS:

$$P(d{:}D) = \sum_{i \in I} \sum_{e_i : E_i} c_i(d, e_i) \rightarrow a_i(f_i(d, e_i)) \cdot P(g_i(d, e_i))$$

$$\boxed{\text{PBES-SRF}} \longrightarrow \boxed{\text{parity game}}$$

state-based

disjunctive:

$$\sigma X(d{:}D) = \bigvee_{i \in I} \exists e_i{:}E_i.\ c_i(d, e_i) \wedge X_i(g_i(d, e_i))$$

conjunctive:

$$\sigma X(d{:}D) = \bigwedge_{i \in I} \forall e_i{:}E_i.\ c_i(d, e_i) \Rightarrow X_i(g_i(d, e_i))$$

Experiments

			basic		opt		pbes-cvc4
PBES	initial node/property	solution	$\|V\|$	time	$\|V\|$	time	time
ball game	winning impossible	false	13	1.04	11/13	0.98	0.27
	infinitely often *put_ball*	true	2	0.01	1/2	0.01	t.o.
train gate	*go(1)* at time 20	true	29	11.55	6/31	5.59	0.39
	fairness	true	19	22.67	5/34	4.91	✗
Fischer (N=3)	no deadlock	true	65	72.90	64/65	62.66	✗
Fischer (N=4)	request must serve	false		o.o.m.	4/38	27.02	✗
bakery	no deadlock	true	23	1.67	23/23	1.80	t.o.
	request must serve	false	123	62.69	13/81	12.97	0.44
Hesselink	cache consistency	false		o.o.m.	20/2461	1849.82	✗
	all writes finish	false		o.o.m.	12/500	27.99	✗
CABP	receive infinitely often	true	260	632.86	25/691	66.44	✗
trading	Xa(1,1)	true	8	0.13	5/12	0.08	t.o.
McCarthy	M(0,10)	true	1633	1299.17	14/419	61.64	✗
	M(0,9)	false	1633	1364.33	116/191	11.89	✗
Takeuchi	T(3,2,1,3)	true		o.o.m.	6/142	50.10	✗
	T(3,2,1,2)	false		o.o.m.	62/159	63.37	✗
ABP+buffer	branching bisimilar	true	132	6.25	131/132	6.45	✗