Distributed Synthesis

Deduktionstreffen 2008

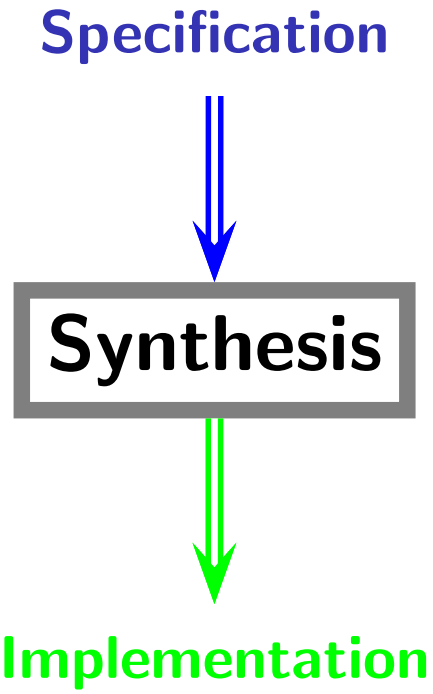
Bernd Finkbeiner

Joint Work with Sven Schewe

Reactive Systems Group
Universität des Saarlandes

March 18, 2008

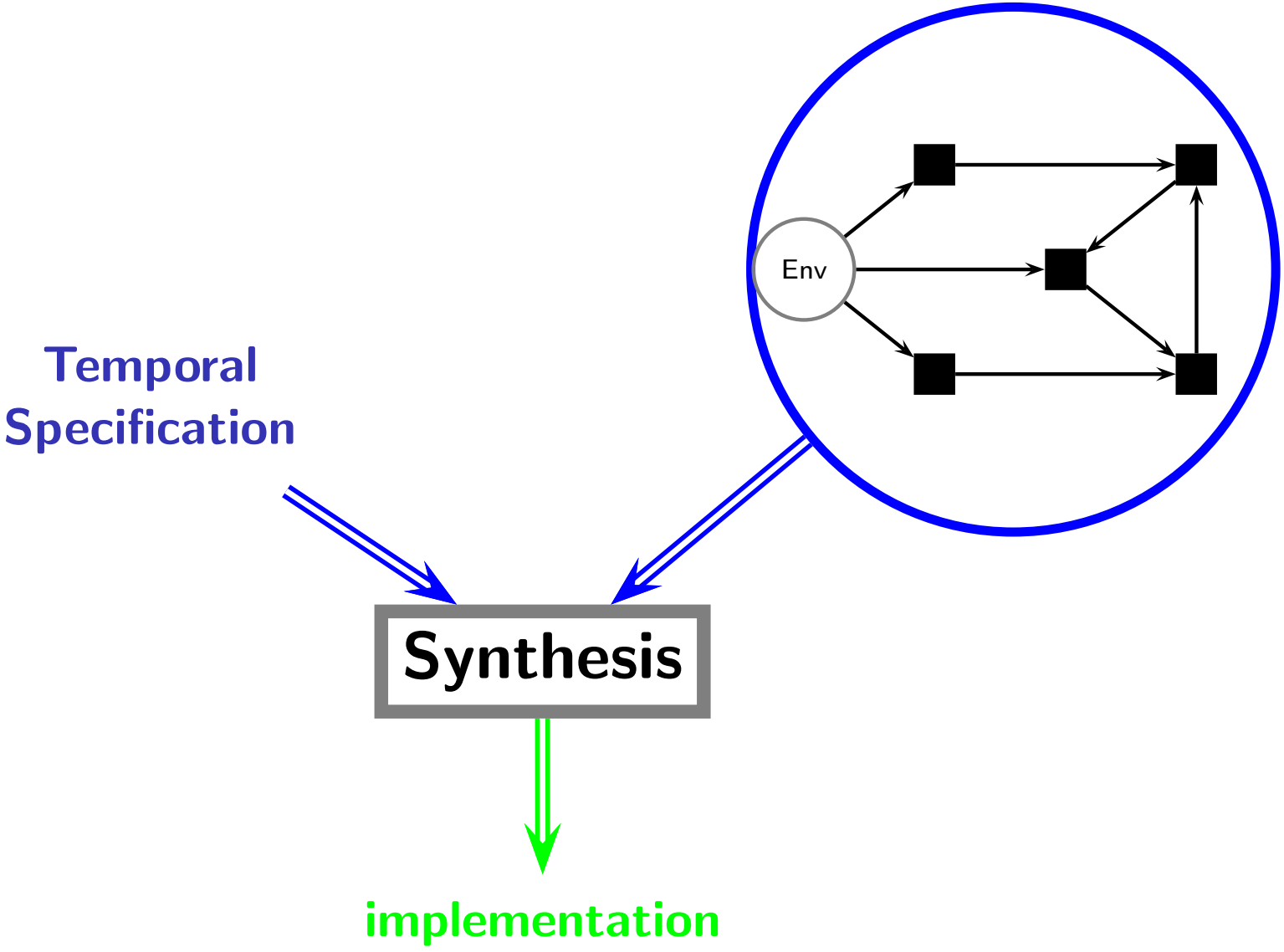
Synthesis



History: (Deductive) Functional Synthesis

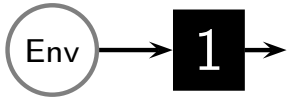
1969: Curry, Howard – Formulas as Types
1977: Burstall, Darlington – Transformations
1980: Manna, Waldinger – Tableau
1982: Mints, Tyugu – Priz
1985: Coquand, Huet – Nuprl
1988: Hayashi, Akano – PX
1990: Smith – KIDS
1990: Abramski – Linear Logic
1992: Murthy – Continuations
1995: Lowry – Amphion
1998: Blaine et al. – Planware
2002: Ellman et al. – Physics-based Animation

Distributed Synthesis



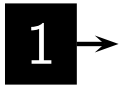
History: (Automata-theoretic) Distributed Synthesis

1962: Church's problem



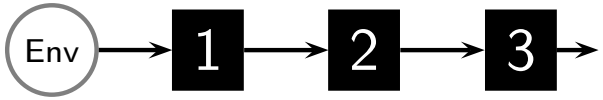
1969: Rabin; Büchi, Landweber – open systems, S1S

1981: Manna, Wolper: closed systems, LTL
Clarke, Emerson: closed systems, CTL

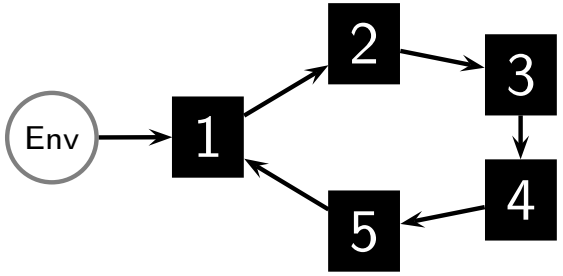


1983: Kozen, Parikh: closed systems, μ-calculus

1990: Pnueli, Rosner: pipeline systems, LTL



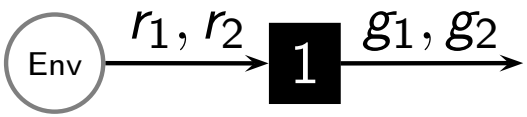
2001: Kupferman, Vardi: one-way rings, CTL*



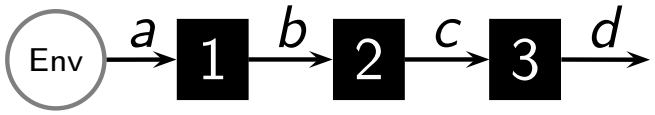
2005: F., Schewe: uniform synthesis

Automata-Theoretic Approaches: Complexity

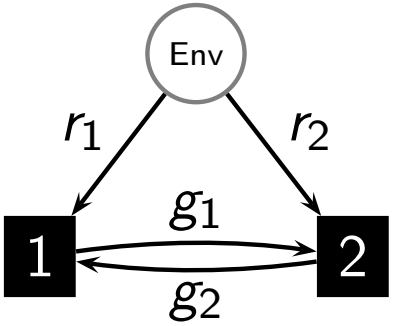
1-process architectures — 2EXPTIME



Pipeline architectures — NONELEMENTARY

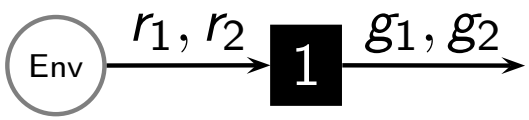


2-process arbiter architecture — UNDECIDABLE

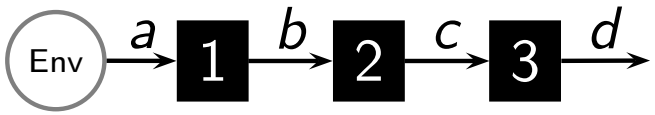


From Input to Output Complexity?

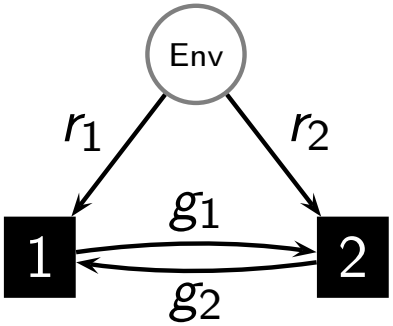
1-process architectures — nondeterministic quasi-linear



Pipeline architectures — nondeterministic quasi-linear



2-process arbiter architecture — nondeterministic quasi-linear



Overview

1-Process Synthesis

specification φ

$\Downarrow$

universal Co-Büchi automaton

automata-theoretic

deterministic parity automaton

$\Downarrow$

parity emptiness check

SMT-based

constraint system

$\Downarrow$

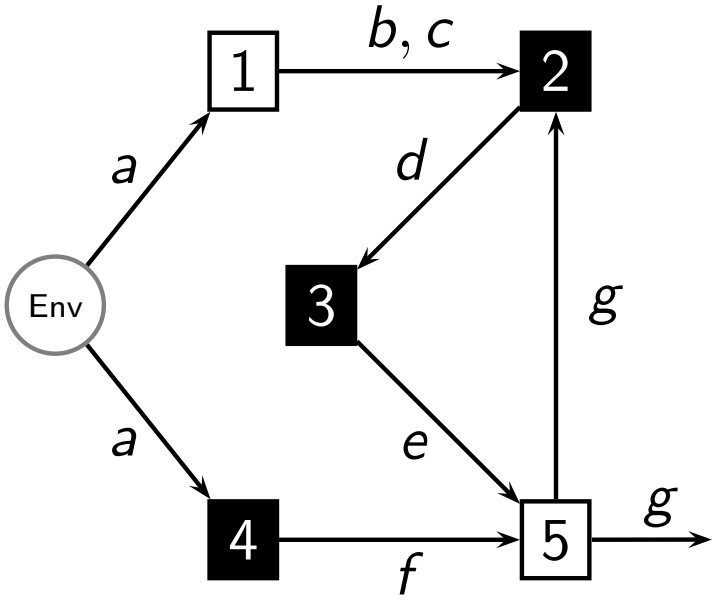
SMT Solving

Multi-Process Synthesis

Sequence of automata transformations (Safra constructions – exponential)

locality constraints

Architecture

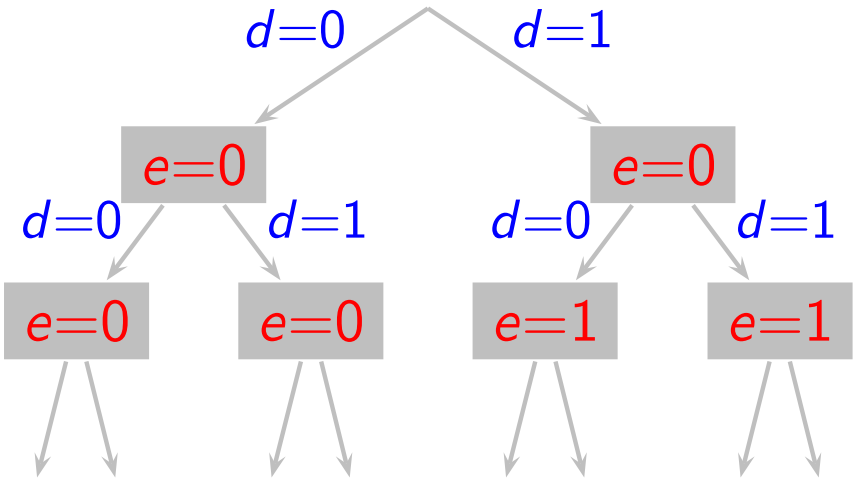


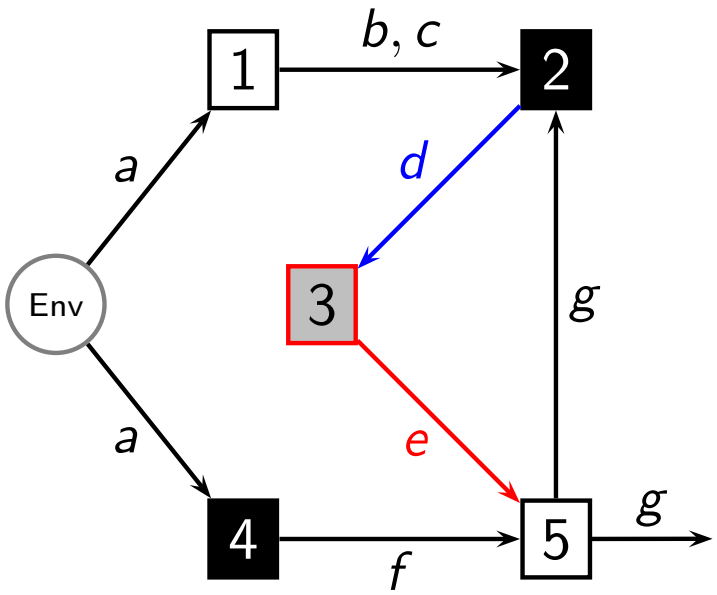
Directed graph:

Nodes:
- white-box processes – **known** implementation
- black-box processes – **unknown** implementation
- environment – unconstrained behavior

Edges:
- communication structure
- variables

Implementation

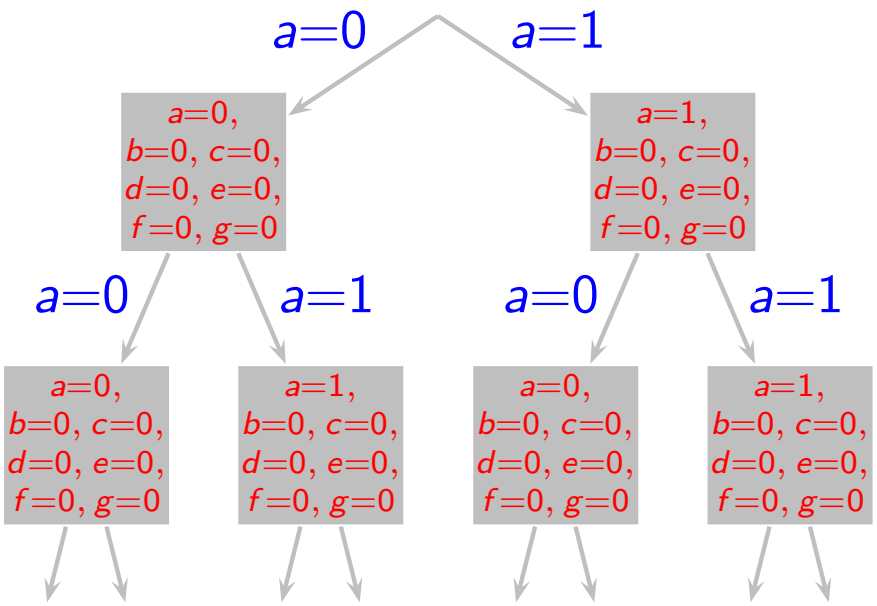


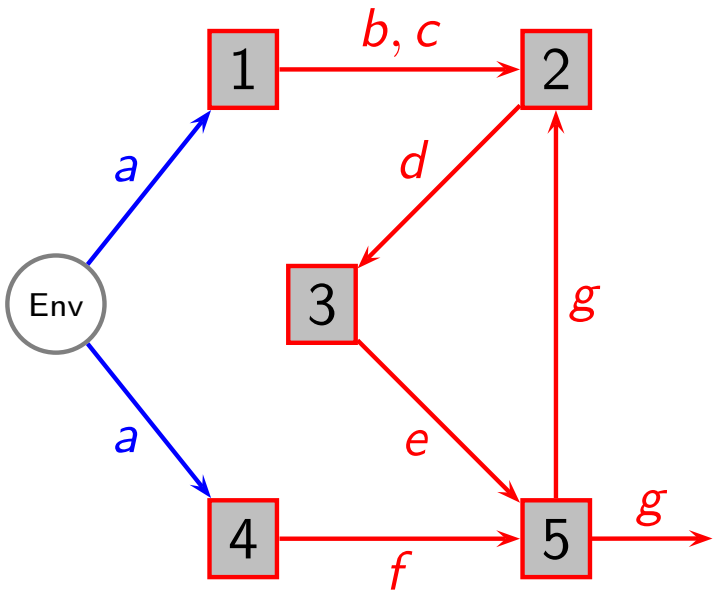


Process implementation

The implementation defines for each process p
with input variables I_p and output variables O_p
a labeled transition system with directions 2^{I_p} and labels 2^{O_p}.

Specification



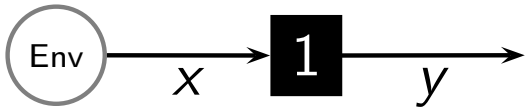


Linear-time Temporal Logic

The combination of the process implementations defines an input-preserving transition system with directions 2^I and labels 2^V, where I are the global input variables and V is the set of variables.

The implementation is correct iff the transition system satisfies the given LTL formula φ.

1-Process Systems: Realizability = Satisfiability

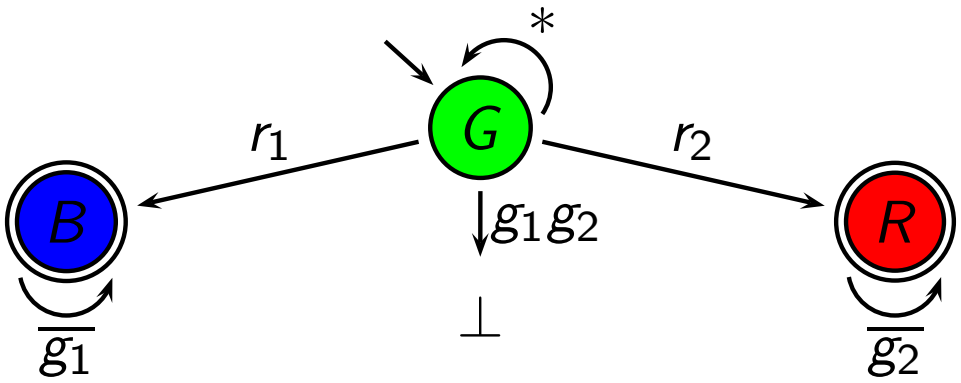


Automata-theoretic approach

Specification φ	$\sim$	automaton $\mathcal{A}_\varphi$
Models of φ	$\sim$	language of $\mathcal{A}_\varphi$
Realizability of φ	$\sim$	language non-emptiness of $\mathcal{A}_\varphi$

- $\mathcal{U}_\varphi$ — universal co-Büchi automaton equivalent to specification φ $\qquad |\mathcal{U}_\varphi| \in \mathsf{EXP}(|\varphi|)$
- $\mathcal{A}_\varphi$ — equivalent deterministic parity automaton $\qquad |\mathcal{A}_\varphi| \in \mathsf{2EXP}(|\varphi|)$
- If $\mathcal{A}_\varphi$ is non-empty, it accepts a correct implementation $\boxed{1}$ with $\boxed{1} \in \mathsf{2EXP}(|\varphi|)$

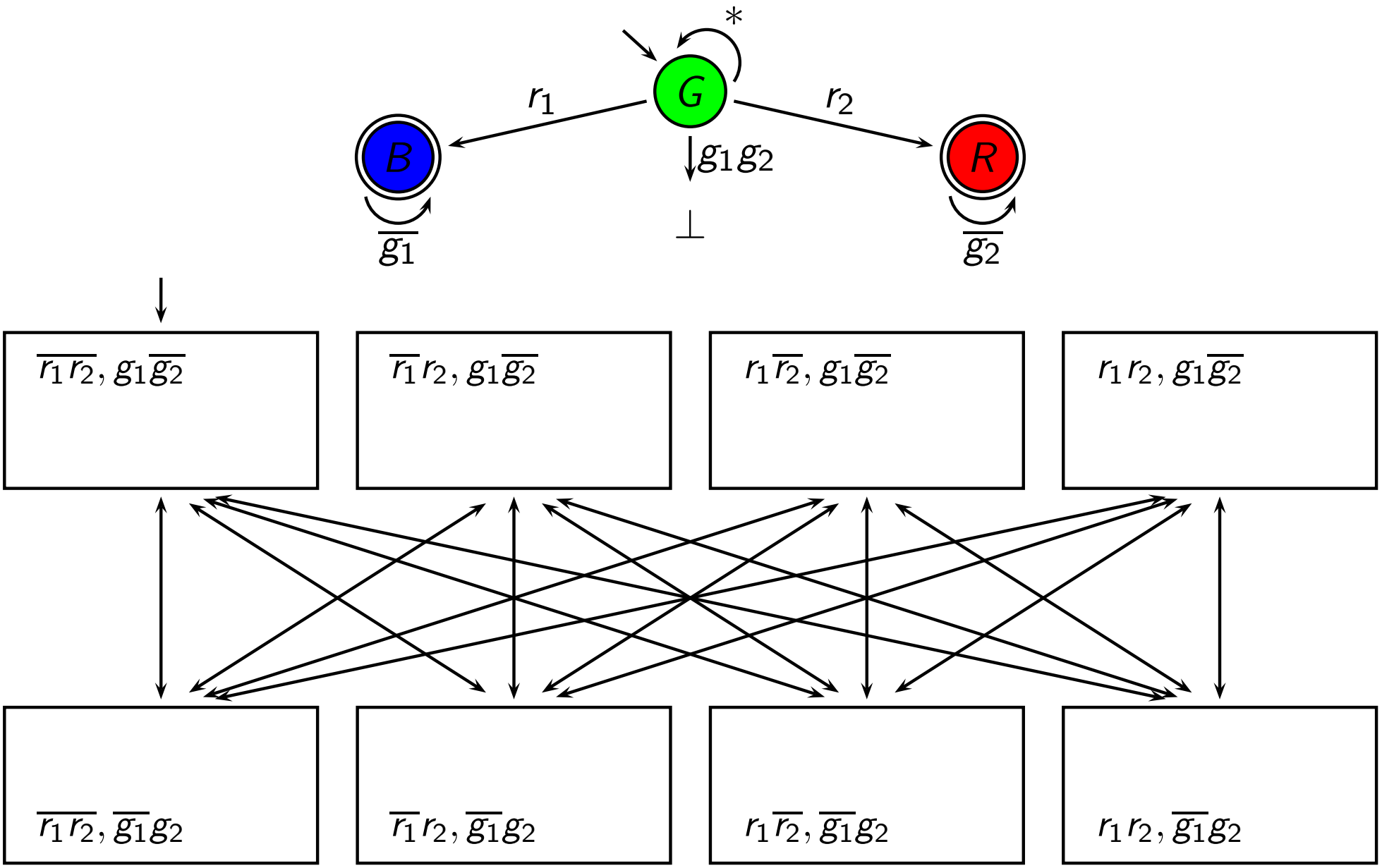
Universal Co-Büchi Automata



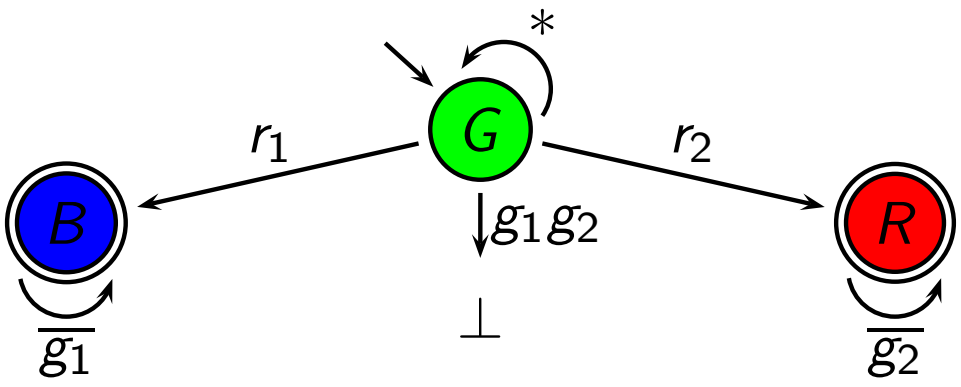
Example: Simplified Arbiter

$\varphi = \Box(r_1 \rightarrow \bigcirc \Diamond g_1) \wedge \Box(r_2 \rightarrow \bigcirc \Diamond g_2) \wedge \neg \Diamond(g_1 \wedge g_2)$

Acceptance of a Transition System



Acceptance of a Transition System



$\overline{r_1}\,\overline{r_2}, g_1\overline{g_2}$

$\overline{r_1}\,r_2, g_1\overline{g_2}$

$r_1\overline{r_2}, g_1\overline{g_2}$

$r_1 r_2, g_1\overline{g_2}$

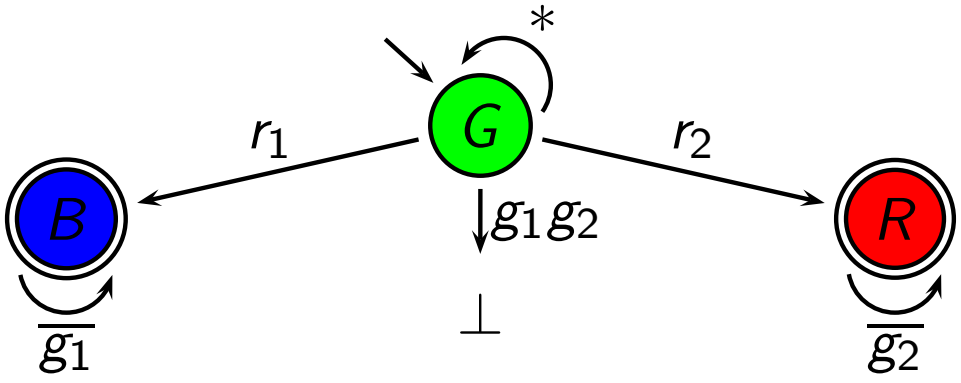
$\overline{r_1}\,\overline{r_2}, \overline{g_1}g_2$

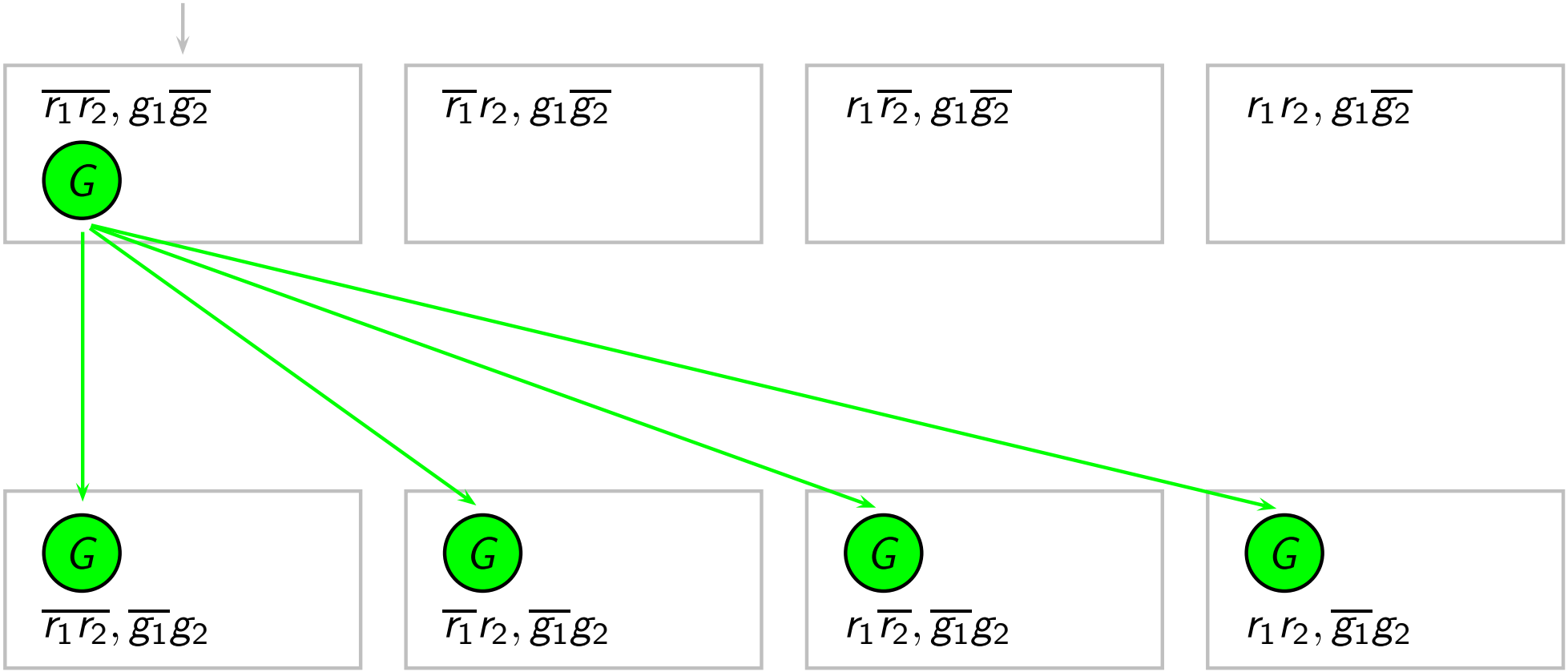
$\overline{r_1}\,r_2, \overline{g_1}g_2$

$r_1\overline{r_2}, \overline{g_1}g_2$

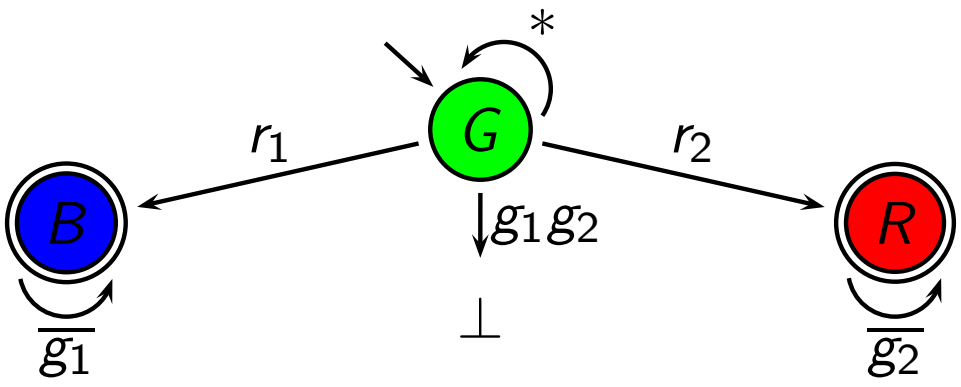
$r_1 r_2, \overline{g_1}g_2$

Acceptance of a Transition System





Acceptance of a Transition System



$\overline{r_1}\,\overline{r_2}, g_1\overline{g_2}$

$\overline{r_1}\,r_2, g_1\overline{g_2}$

$r_1\overline{r_2}, g_1\overline{g_2}$

$r_1 r_2, g_1\overline{g_2}$

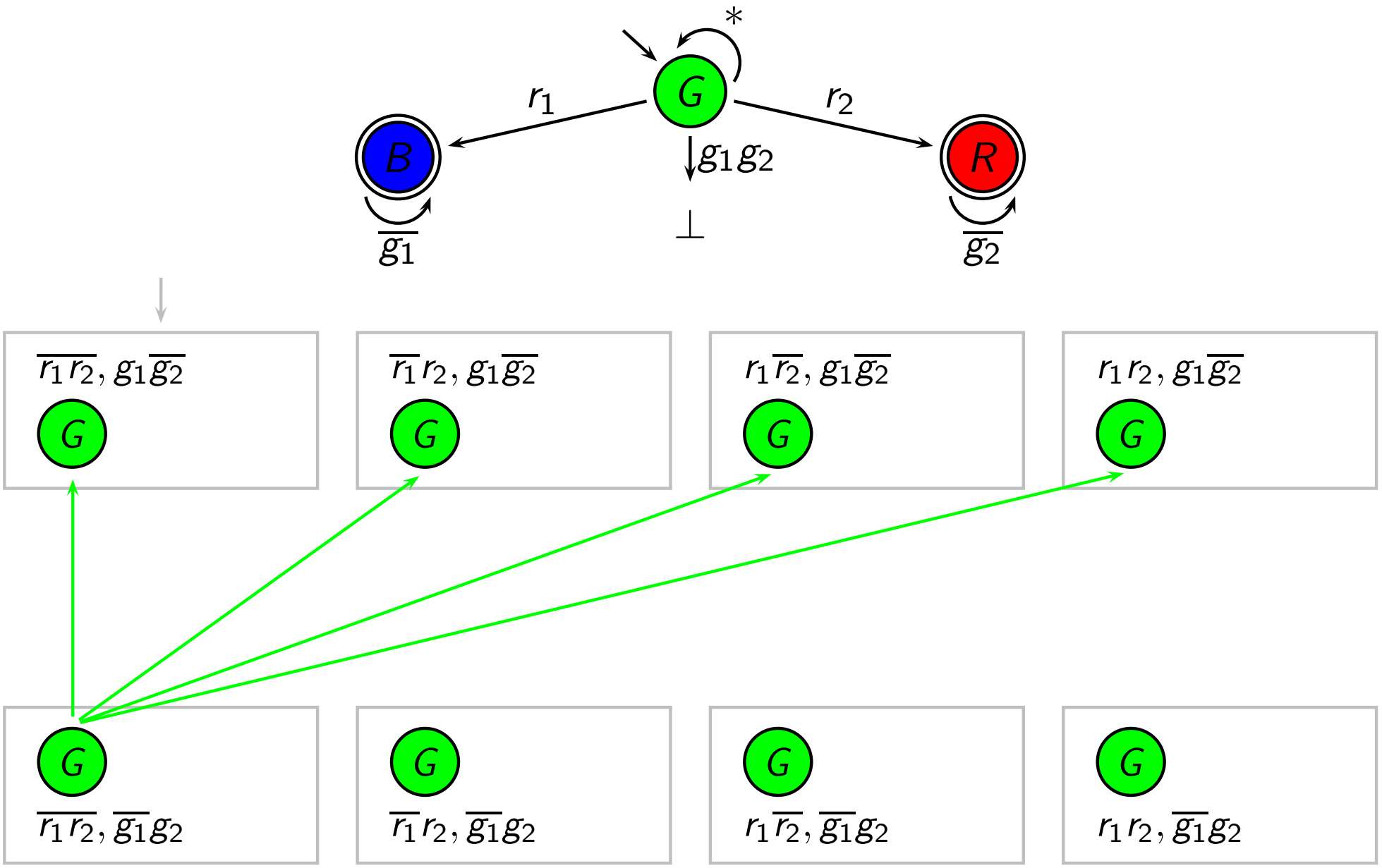
G
$\overline{r_1}\,\overline{r_2}, \overline{g_1}g_2$

G
$\overline{r_1}\,r_2, \overline{g_1}g_2$

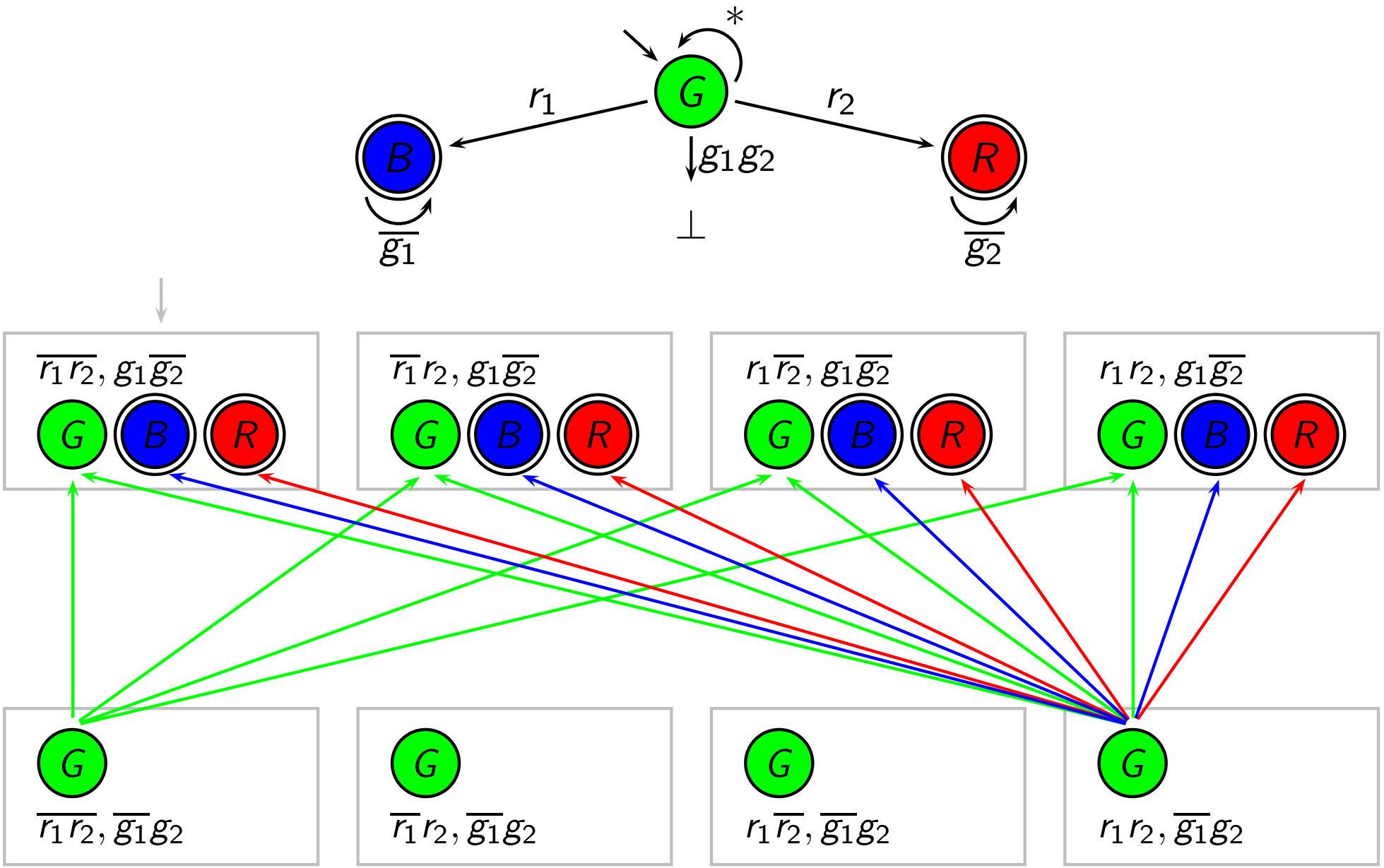
G
$r_1\overline{r_2}, \overline{g_1}g_2$

G
$r_1 r_2, \overline{g_1}g_2$

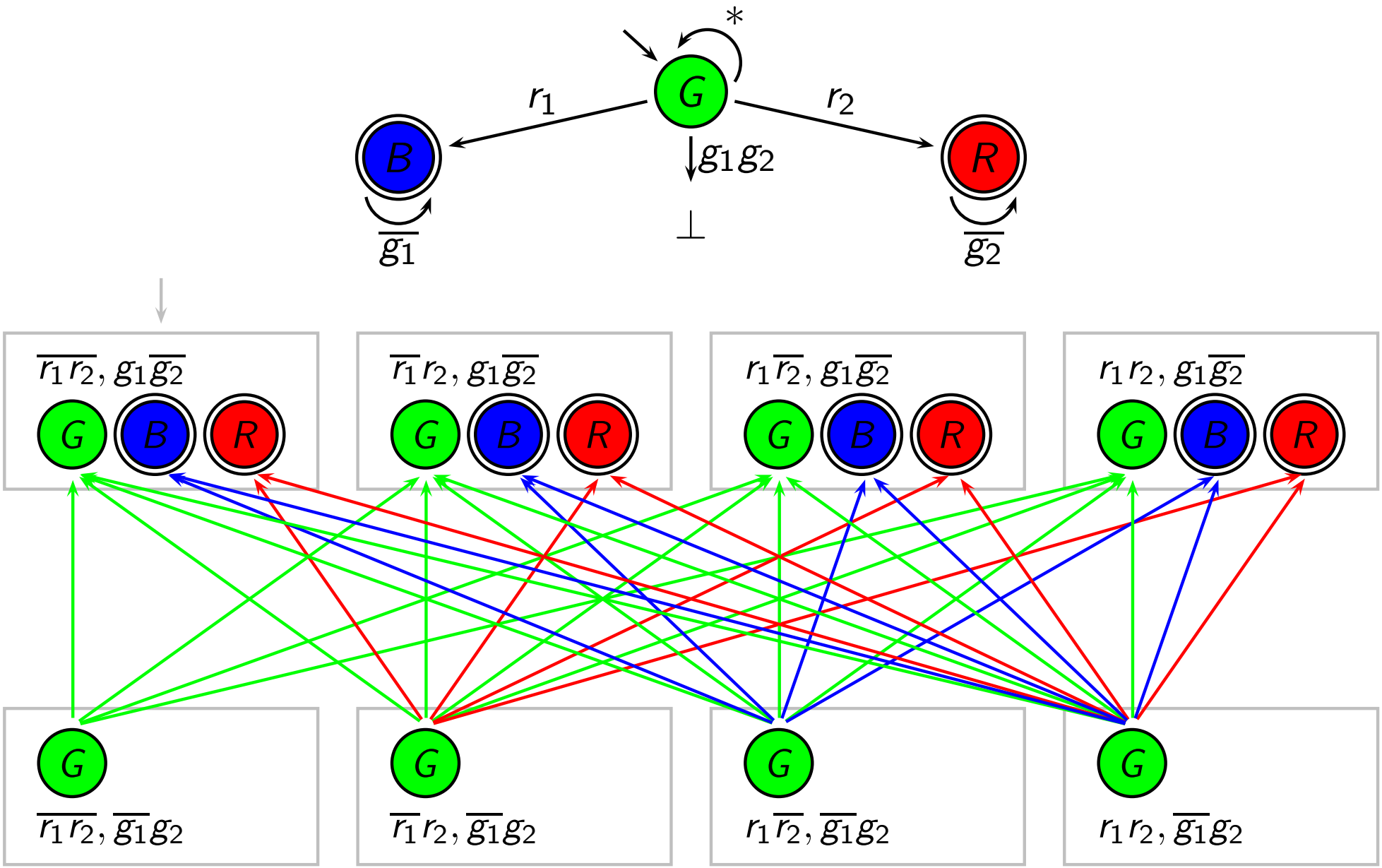
Acceptance of a Transition System



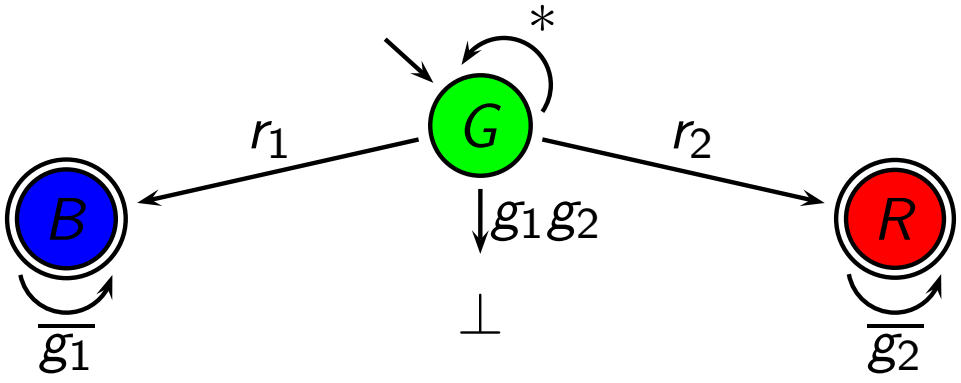
Acceptance of a Transition System

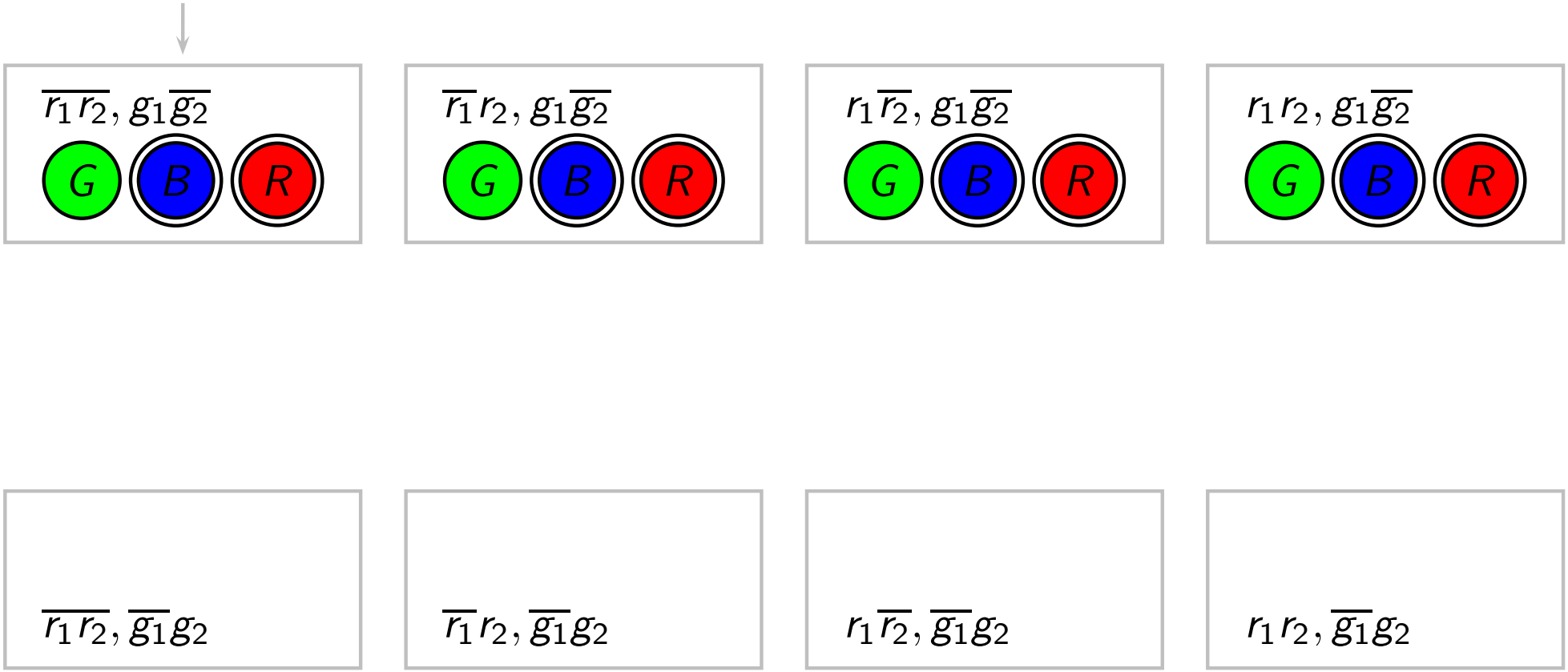


Acceptance of a Transition System

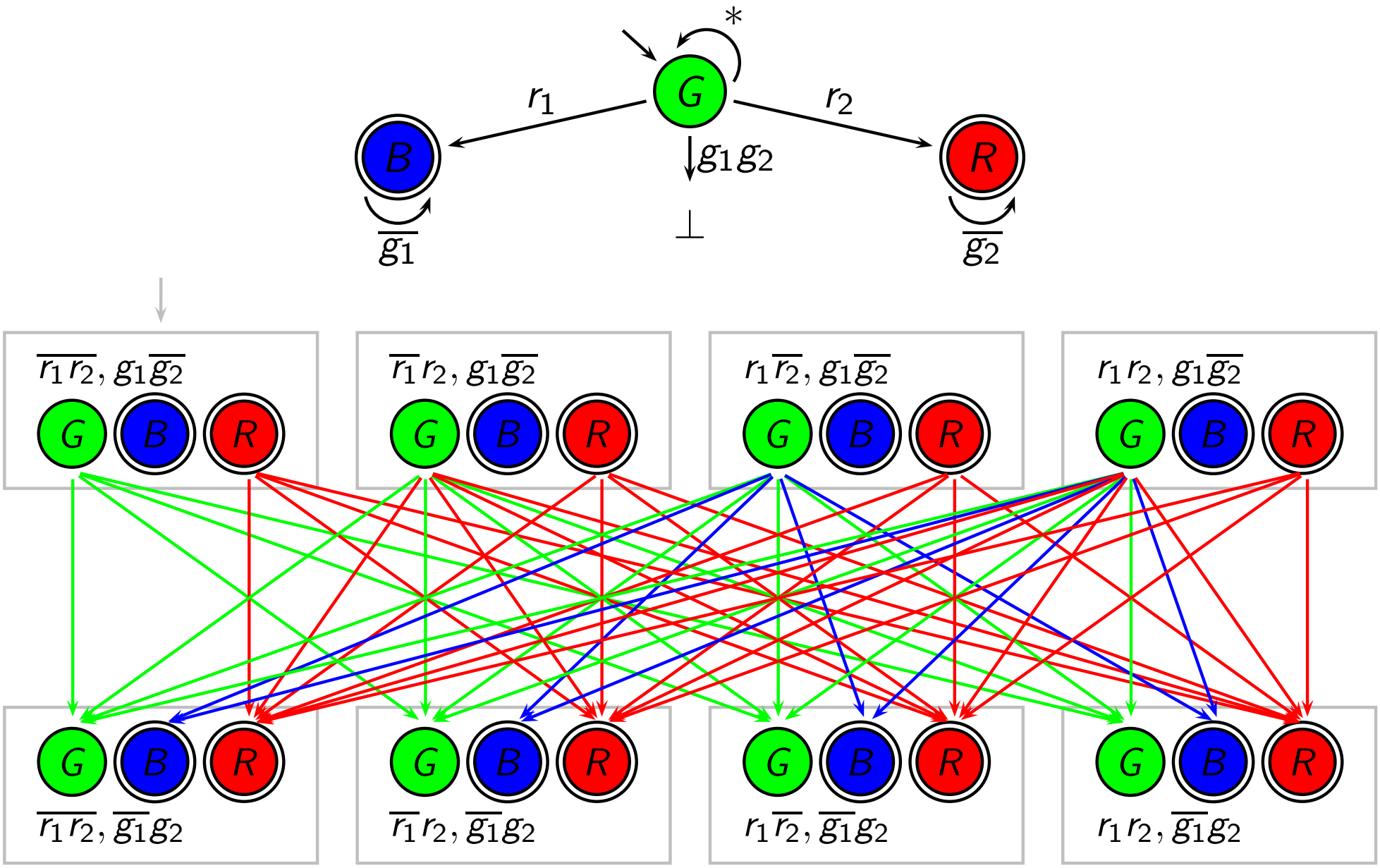


Acceptance of a Transition System

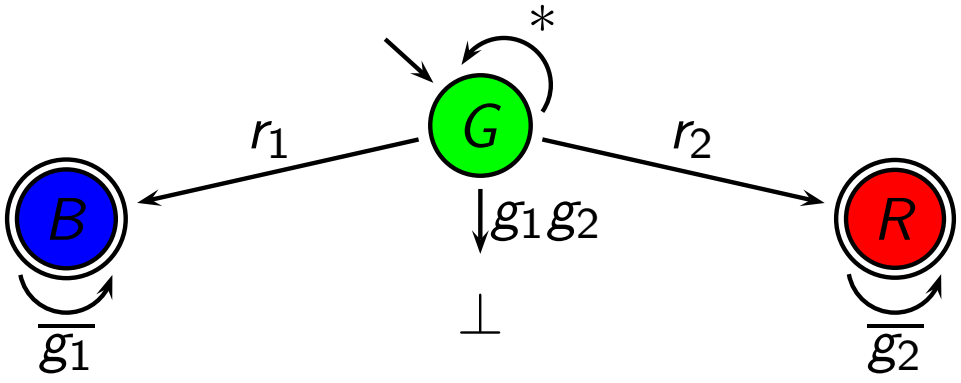




Acceptance of a Transition System



Acceptance of a Transition System

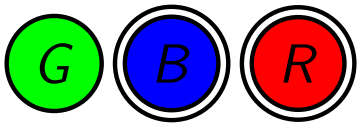


$\overline{r_1}\,\overline{r_2}, g_1\overline{g_2}$

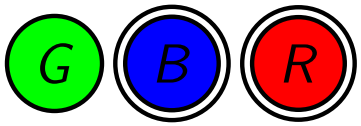
$\overline{r_1}\,r_2, g_1\overline{g_2}$

$r_1\overline{r_2}, g_1\overline{g_2}$

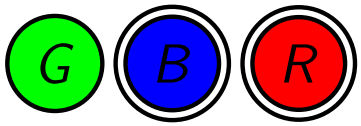
$r_1 r_2, g_1\overline{g_2}$



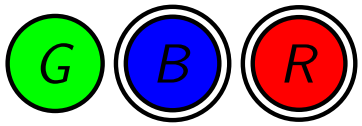
$\overline{r_1}\,\overline{r_2}, \overline{g_1}g_2$



$\overline{r_1}\,r_2, \overline{g_1}g_2$

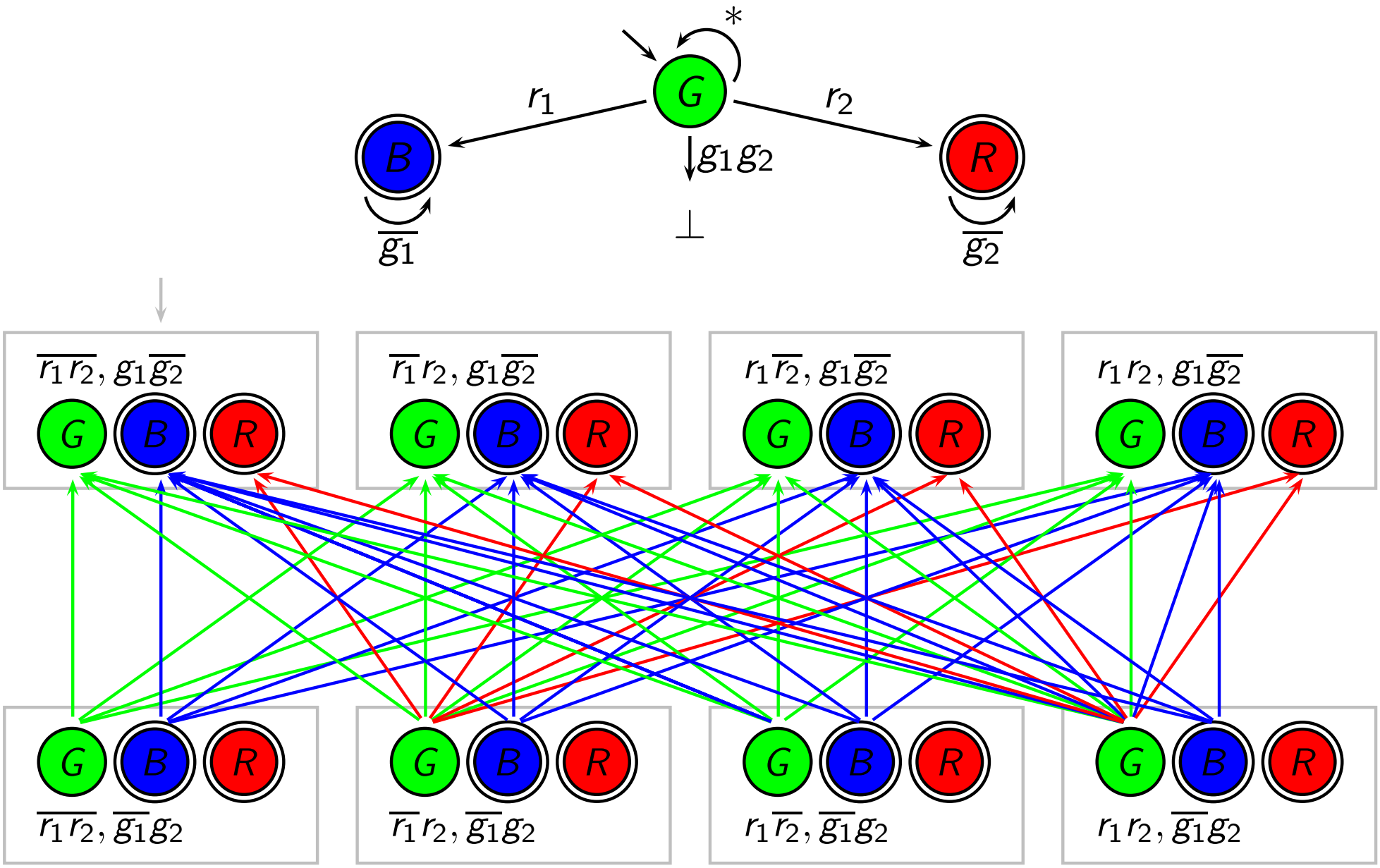


$r_1\overline{r_2}, \overline{g_1}g_2$

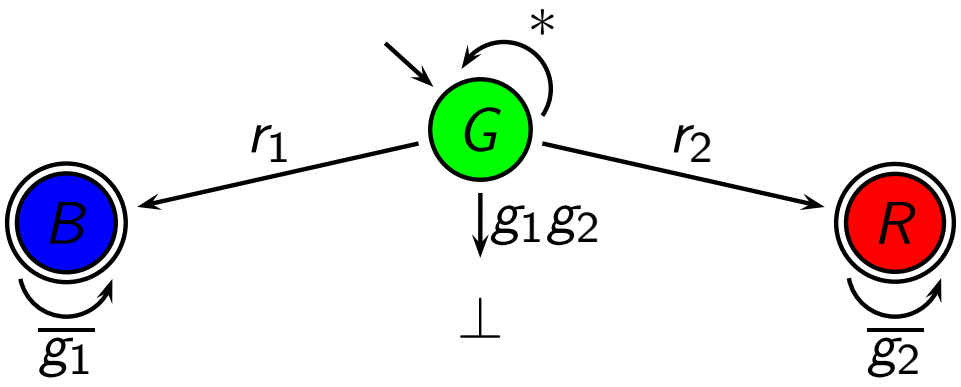


$r_1 r_2, \overline{g_1}g_2$

Acceptance of a Transition System

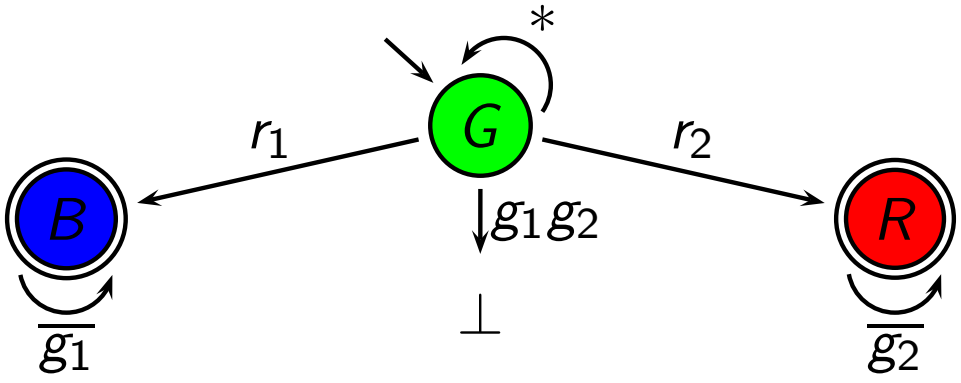


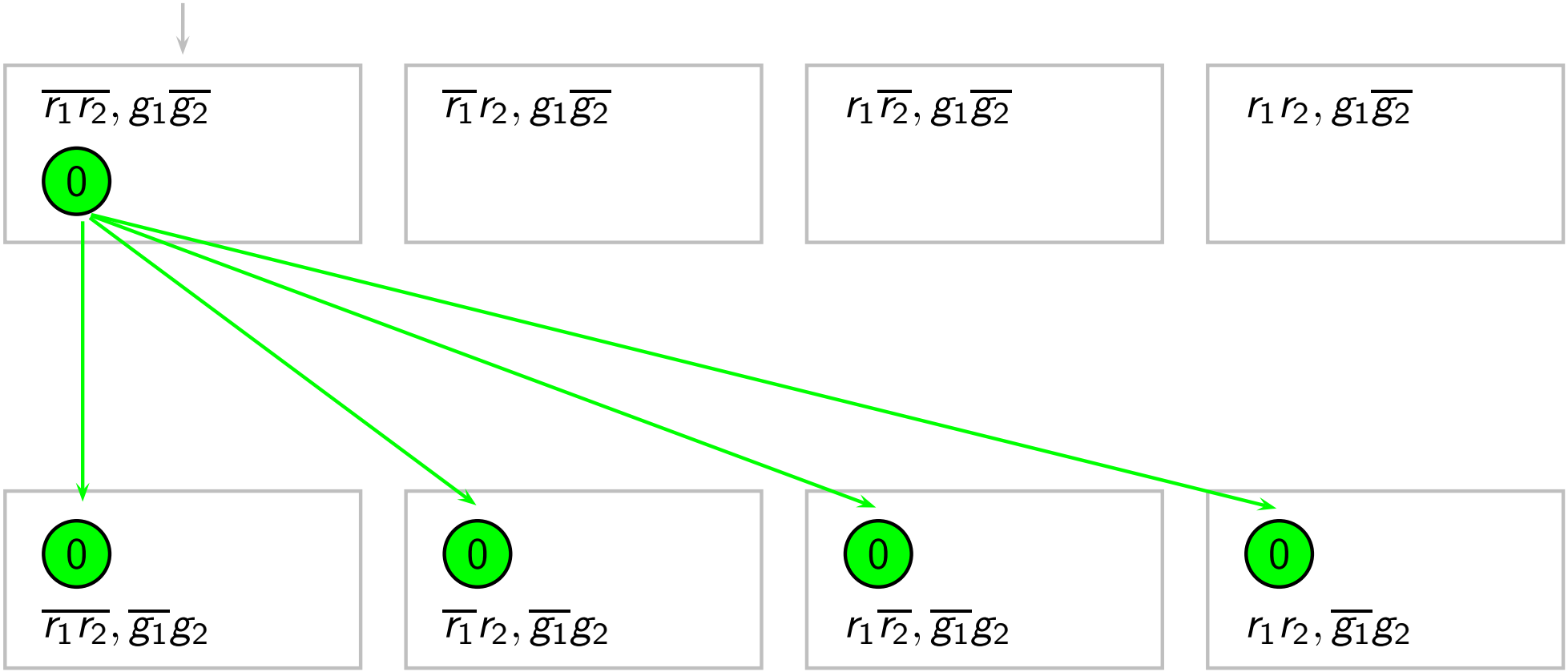
Acceptance of a Transition System



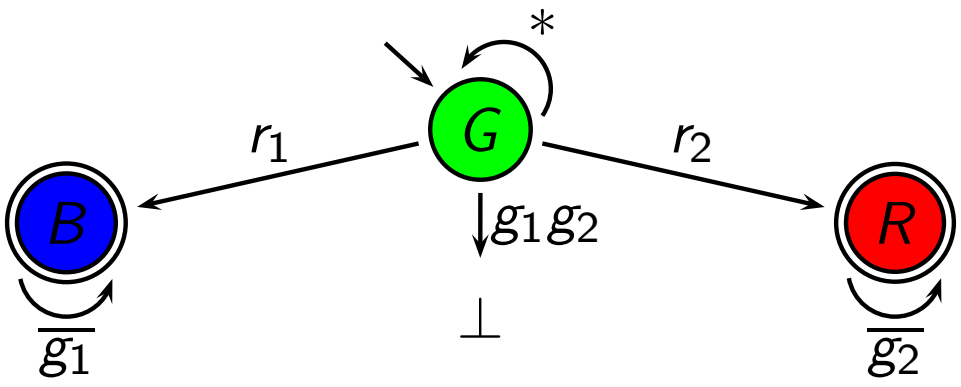
$\overline{r_1}\,\overline{r_2}, g_1\overline{g_2}$ 	$\overline{r_1}\,r_2, g_1\overline{g_2}$	$r_1\overline{r_2}, g_1\overline{g_2}$	$r_1 r_2, g_1\overline{g_2}$
$\overline{r_1}\,\overline{r_2}, \overline{g_1}g_2$	$\overline{r_1}\,r_2, \overline{g_1}g_2$	$r_1\overline{r_2}, \overline{g_1}g_2$	$r_1 r_2, \overline{g_1}g_2$

Acceptance of a Transition System



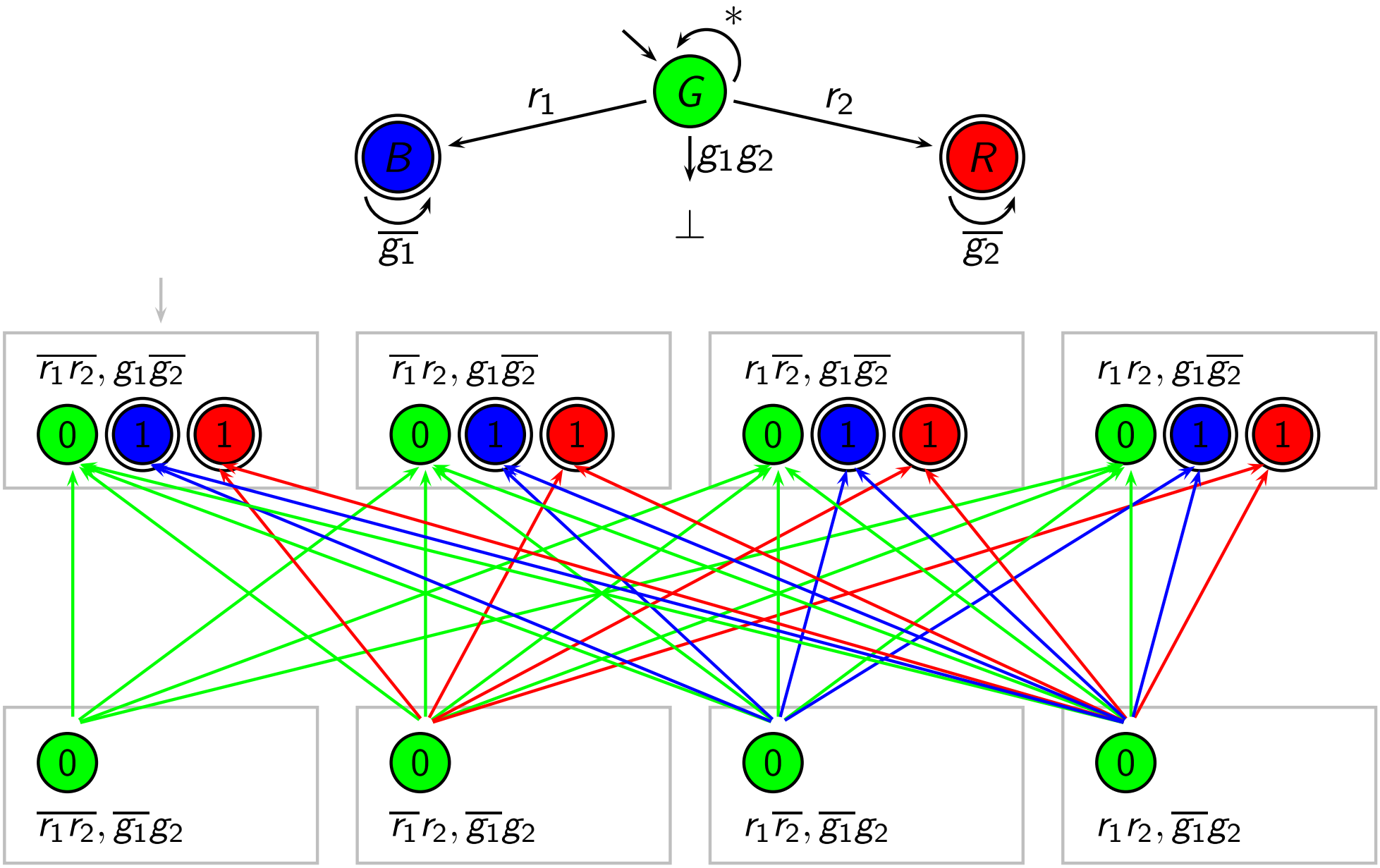


Acceptance of a Transition System

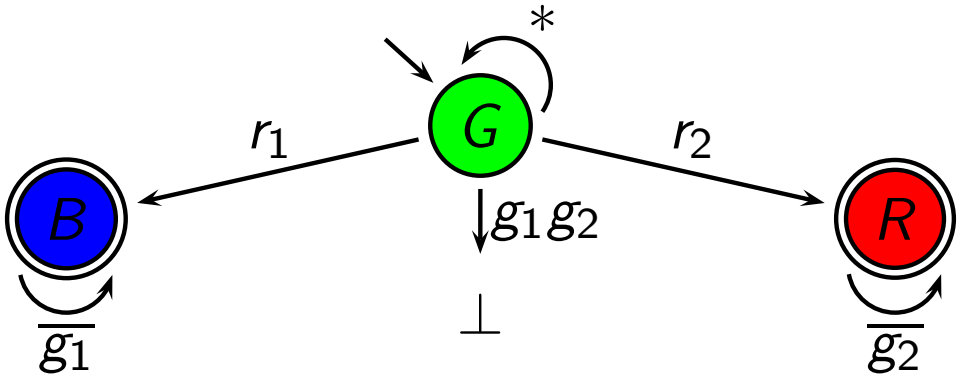


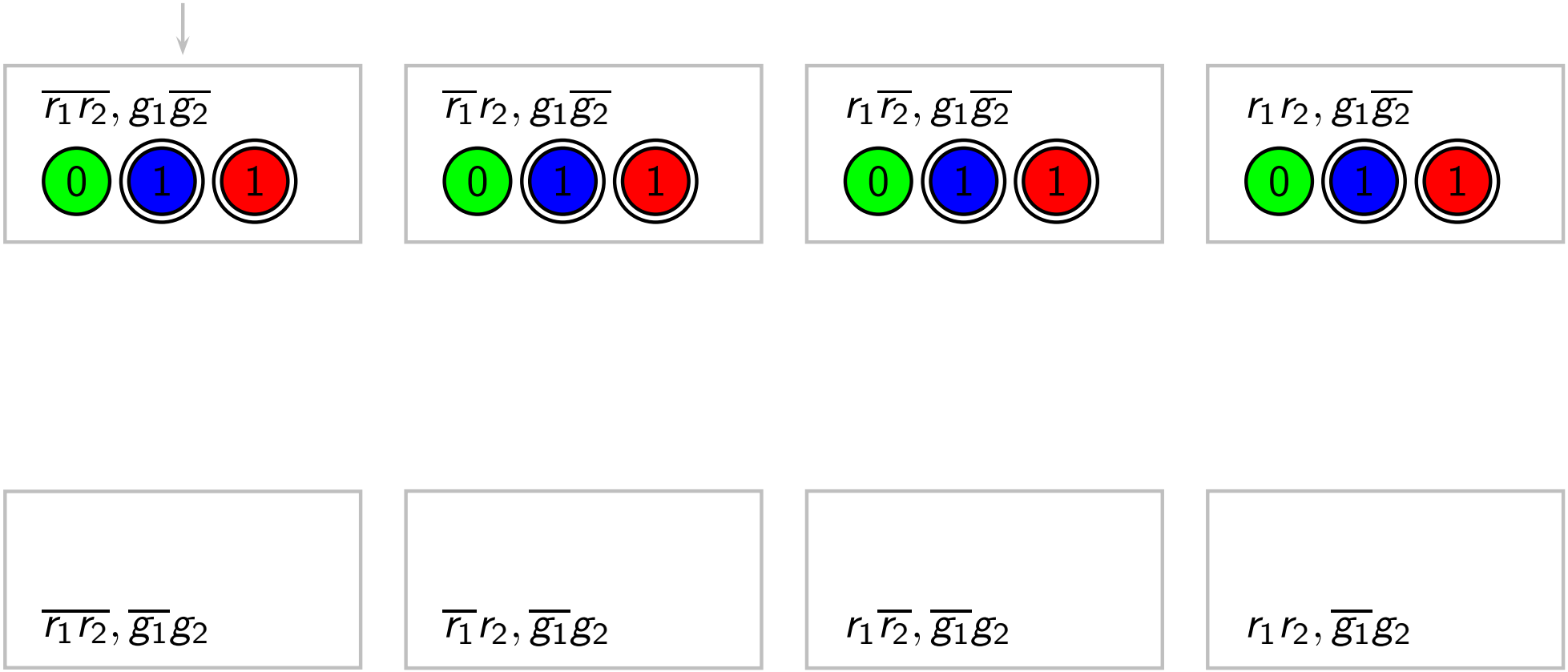
$\overline{r_1}\,\overline{r_2}, g_1\overline{g_2}$	$\overline{r_1}\,r_2, g_1\overline{g_2}$	$r_1\overline{r_2}, g_1\overline{g_2}$	$r_1 r_2, g_1\overline{g_2}$
0 $\overline{r_1}\,\overline{r_2}, \overline{g_1}g_2$	0 $\overline{r_1}\,r_2, \overline{g_1}g_2$	0 $r_1\overline{r_2}, \overline{g_1}g_2$	0 $r_1 r_2, \overline{g_1}g_2$

Acceptance of a Transition System

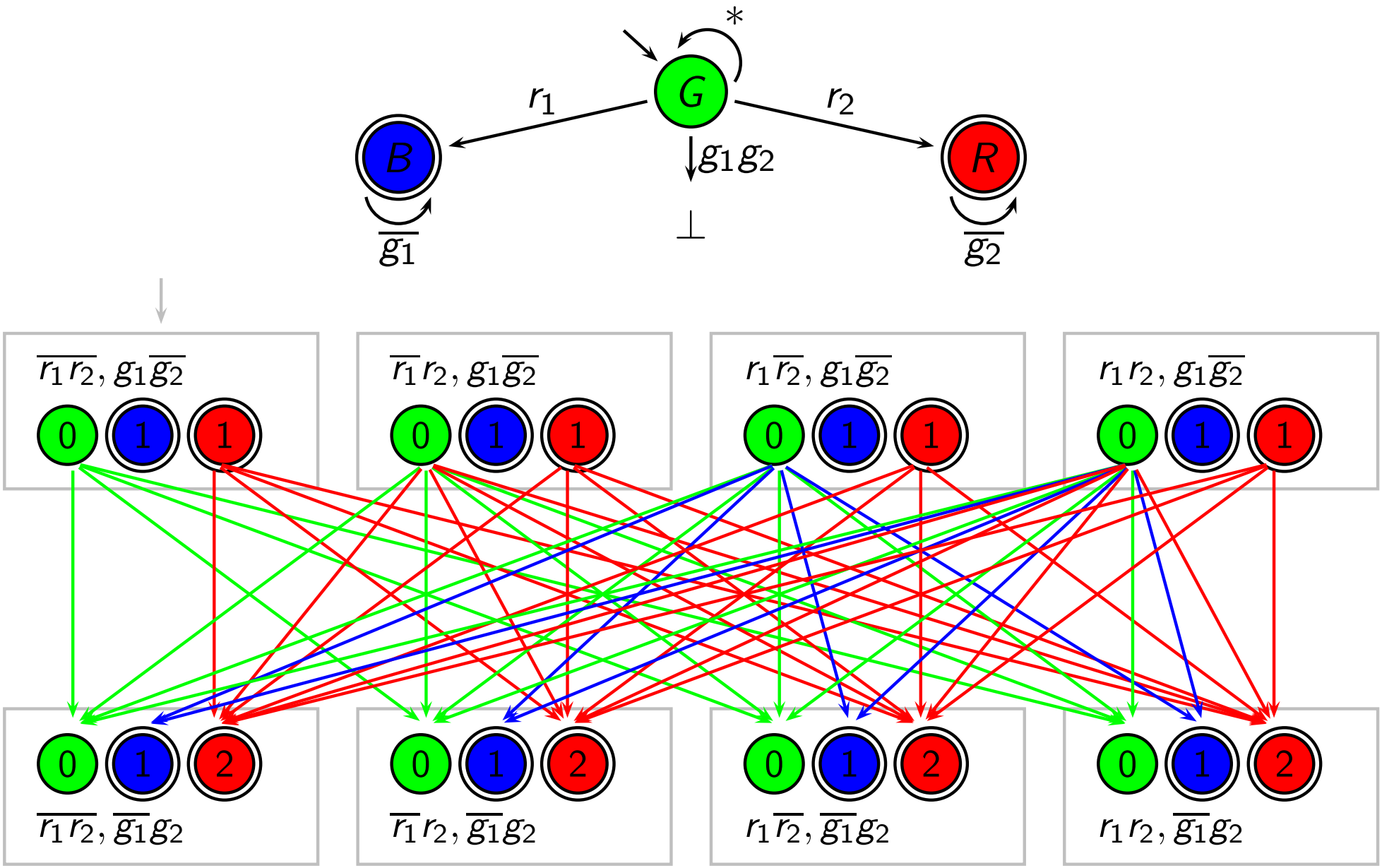


Acceptance of a Transition System

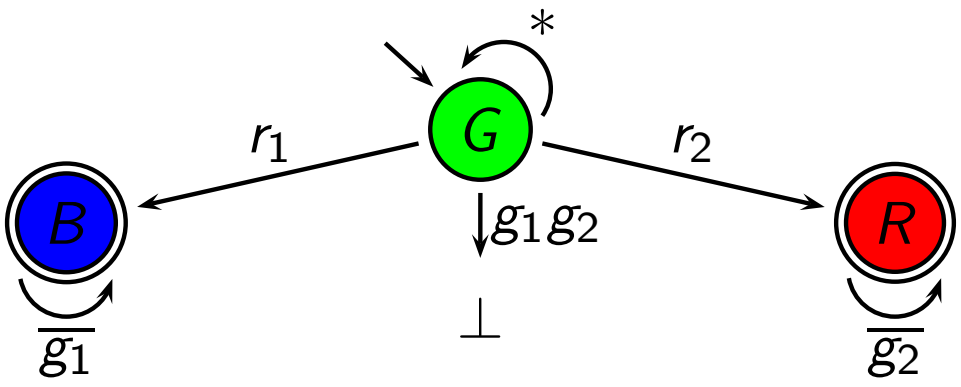




Acceptance of a Transition System



Acceptance of a Transition System

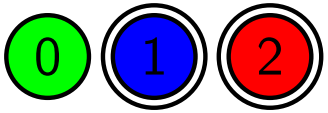


$\overline{r_1}\,\overline{r_2}, g_1\overline{g_2}$

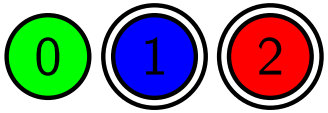
$\overline{r_1}\,r_2, g_1\overline{g_2}$

$r_1\overline{r_2}, g_1\overline{g_2}$

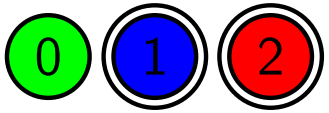
$r_1 r_2, g_1\overline{g_2}$



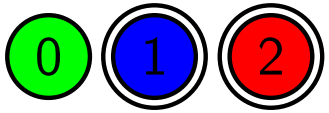
$\overline{r_1}\,\overline{r_2}, \overline{g_1}g_2$



$\overline{r_1}\,r_2, \overline{g_1}g_2$

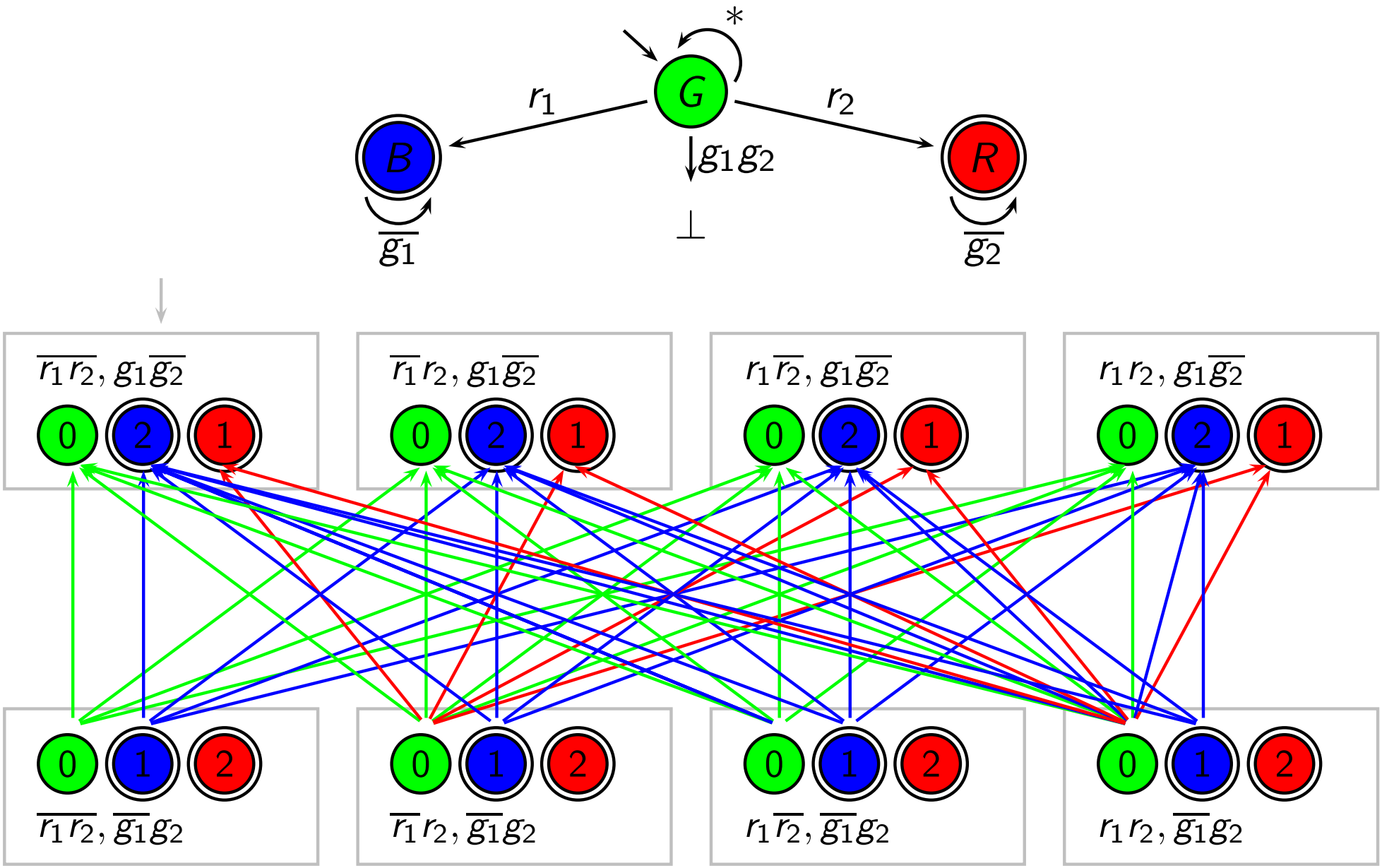


$r_1\overline{r_2}, \overline{g_1}g_2$



$r_1 r_2, \overline{g_1}g_2$

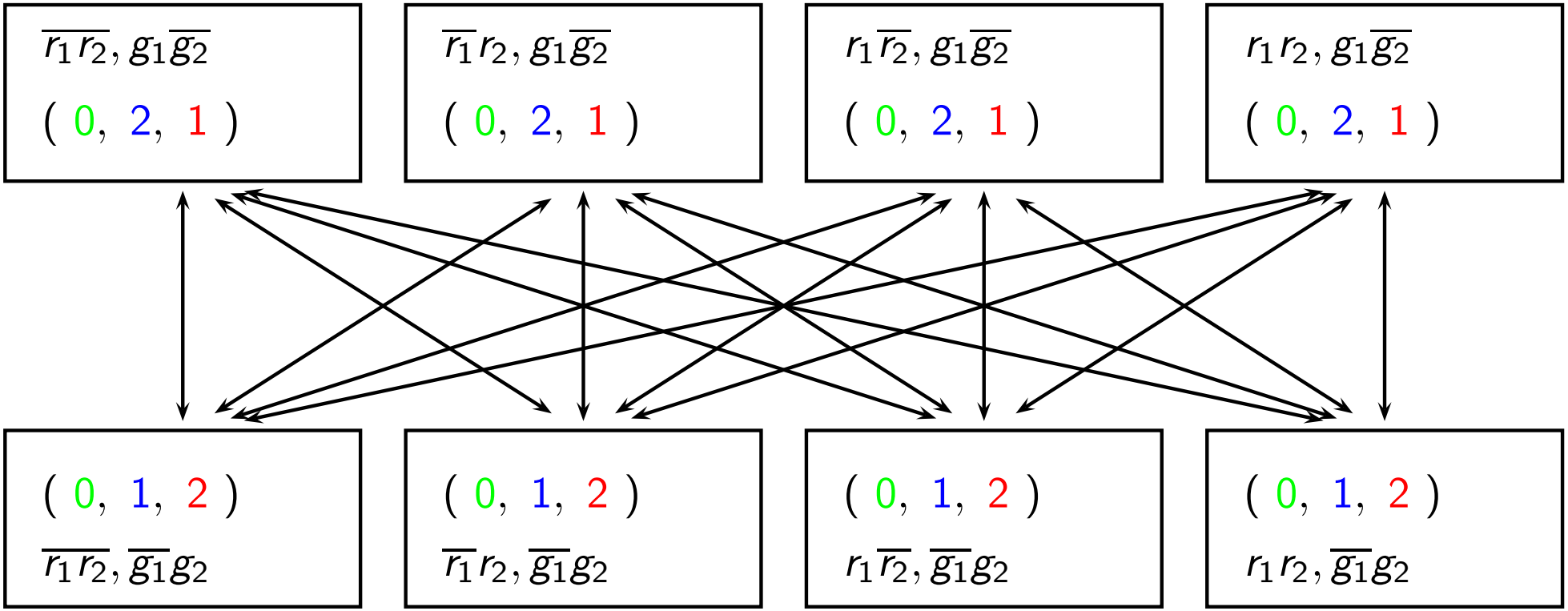
Acceptance of a Transition System



SMT-based Approach: Annotated Transition System

Annotation

- collects the paths of the run tree that lead to a state in the transition system
- for each automaton state, indicates whether state visited on some path, and if so, max number of visits to rejecting states



SMT-based Approach: Annotated Transition System

Annotation

- collects the paths of the run tree that lead to a state in the transition system
- for each automaton state, indicates whether state visited on some path, and if so, max number of visits to rejecting states

Theorem

An (input preserving) transition system is accepted by a $\mathcal{UCB}$
$\Leftrightarrow$ it has a valid annotation.

Proof idea

Cycle with rejecting state reachable in the run graph
$\Leftrightarrow$ no valid annotation.

SMT-based Approach: Constraint System

The constraint system specifies the existence of an annotated transition system.

Representation of transition system

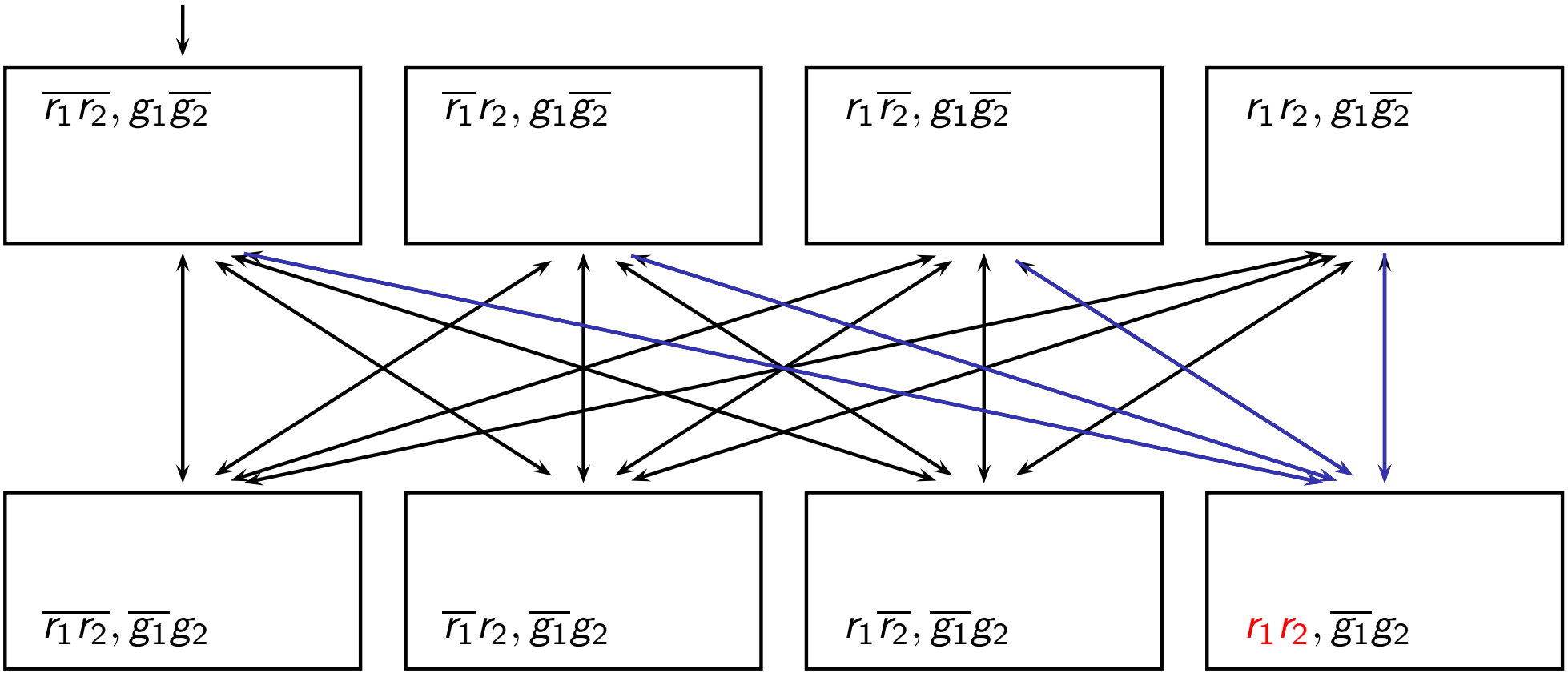
- states: $\mathbb{N}_N$
- labeling: functions $v : \mathbb{N}_N \to \mathbb{B}$
- transitions: functions $\tau_{in} : \mathbb{N}_N \to \mathbb{N}_N$

Representation of annotation

- state occurrence: functions $\lambda_q^{\mathbb{B}} : \mathbb{N}_N \to \mathbb{B}$
- rejecting bound: functions $\lambda_q^{\#} : \mathbb{N}_N \to \mathbb{N}$

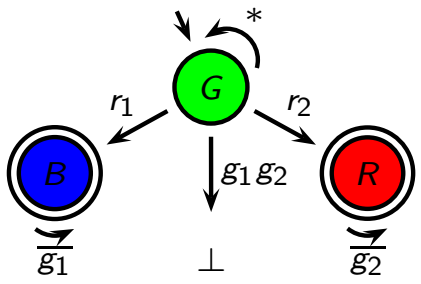
SMT-based Approach: Constraints

$$\forall t.\ r_1(\tau_{r_1 r_2}(t)) \wedge r_2(\tau_{r_1 r_2}(t)) \wedge r_1(\tau_{r_1 \overline{r}_2}(t)) \wedge \neg r_2(\tau_{r_1 \overline{r}_2}(t))$$
$$\wedge\ \neg r_1(\tau_{\overline{r}_1 r_2}(t)) \wedge r_2(\tau_{\overline{r}_1 r_2}(t)) \wedge \neg r_1(\tau_{\overline{r}_1 \overline{r}_2}(t)) \wedge \neg r_2(\tau_{\overline{r}_1 \overline{r}_2}(t))$$



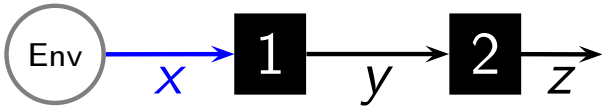
SMT-based Approach: Constraints

- $\lambda_G^{\mathbb{B}}(0)$
- $\forall t.\, \lambda_G^{\mathbb{B}}(t) \rightarrow \lambda_G^{\mathbb{B}}(\tau_{\overline{r}_1\overline{r}_2}(t)) \wedge \lambda_G^{\#}(\tau_{\overline{r}_1\overline{r}_2}(t)) \geq \lambda_G^{\#}(t)$
  $\wedge\ \lambda_G^{\mathbb{B}}(\tau_{\overline{r}_1 r_2}(t)) \wedge \lambda_G^{\#}(\tau_{\overline{r}_1 r_2}(t)) \geq \lambda_G^{\#}(t)$
  $\wedge\ \lambda_G^{\mathbb{B}}(\tau_{r_1\overline{r}_2}(t)) \wedge \lambda_G^{\#}(\tau_{r_1\overline{r}_2}(t)) \geq \lambda_G^{\#}(t)$
  $\wedge\ \lambda_G^{\mathbb{B}}(\tau_{r_1 r_2}(t)) \wedge \lambda_G^{\#}(\tau_{r_1 r_2}(t)) \geq \lambda_G^{\#}(t)$



- $\forall t.\, \lambda_G^{\mathbb{B}}(t) \rightarrow \neg g_1(t) \vee \neg g_2(t)$
- $\forall t.\, \lambda_G^{\mathbb{B}}(t) \wedge r_1(t) \rightarrow \lambda_B^{\mathbb{B}}(\tau_{\overline{r}_1\overline{r}_2}(t)) \wedge \lambda_B^{\#}(\tau_{\overline{r}_1\overline{r}_2}(t)) > \lambda_G^{\#}(t)$
  $\wedge\ \lambda_B^{\mathbb{B}}(\tau_{\overline{r}_1 r_2}(t)) \wedge \lambda_B^{\#}(\tau_{\overline{r}_1 r_2}(t)) > \lambda_G^{\#}(t)$
  $\wedge\ \lambda_B^{\mathbb{B}}(\tau_{r_1\overline{r}_2}(t)) \wedge \lambda_B^{\#}(\tau_{r_1\overline{r}_2}(t)) > \lambda_G^{\#}(t)$
  $\wedge\ \lambda_B^{\mathbb{B}}(\tau_{r_1 r_2}(t)) \wedge \lambda_B^{\#}(\tau_{r_1 r_2}(t)) > \lambda_G^{\#}(t)$
- $\forall t.\, \lambda_B^{\mathbb{B}}(t) \wedge \neg g_1(t) \rightarrow \lambda_B^{\mathbb{B}}(\tau_{\overline{r}_1\overline{r}_2}(t)) \wedge \lambda_B^{\#}(\tau_{\overline{r}_1\overline{r}_2}(t)) > \lambda_B^{\#}(t)$
  $\wedge\ \lambda_B^{\mathbb{B}}(\tau_{\overline{r}_1 r_2}(t)) \wedge \lambda_B^{\#}(\tau_{\overline{r}_1 r_2}(t)) > \lambda_B^{\#}(t)$
  $\wedge\ \lambda_B^{\mathbb{B}}(\tau_{r_1\overline{r}_2}(t)) \wedge \lambda_B^{\#}(\tau_{r_1\overline{r}_2}(t)) > \lambda_B^{\#}(t)$
  $\wedge\ \lambda_B^{\mathbb{B}}(\tau_{r_1 r_2}(t)) \wedge \lambda_B^{\#}(\tau_{r_1 r_2}(t)) > \lambda_B^{\#}(t)$

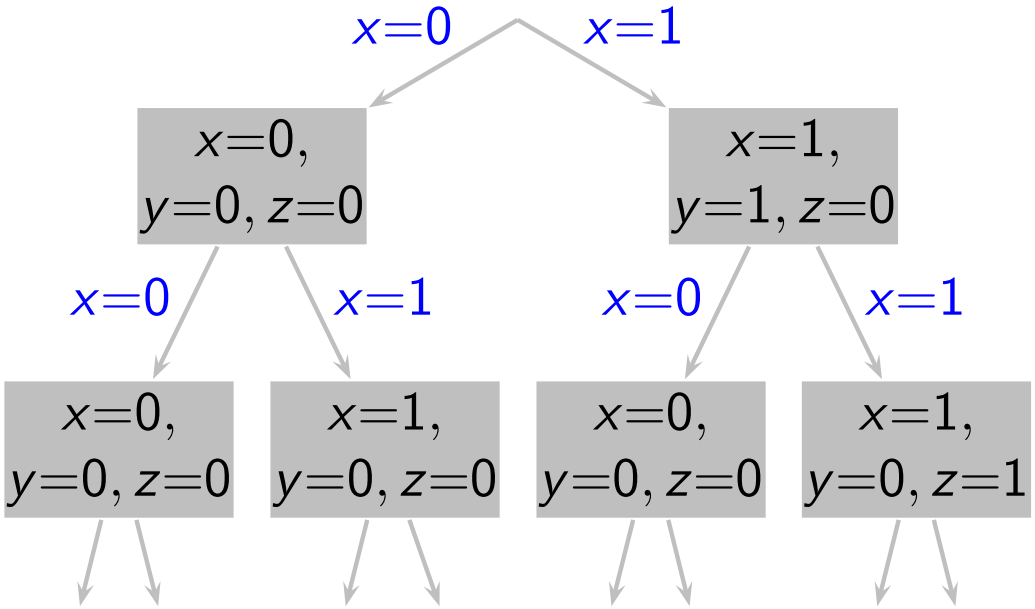
Multi-Process Systems: Realizability $\neq$ Satisfiability



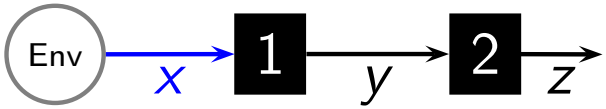
Realizability $\neq$ Satisfiability

reason: incomplete information
process 2 does not know x.

decisions of process 2
must not depend on x.



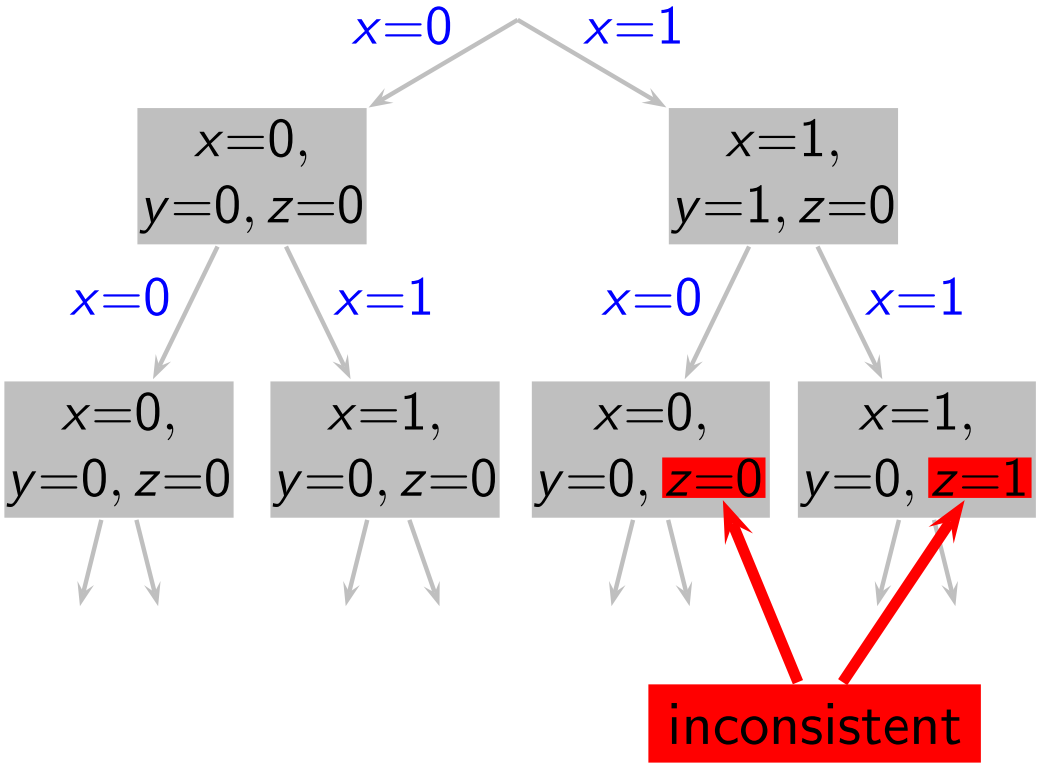
Synthesis of Multi-Process Systems



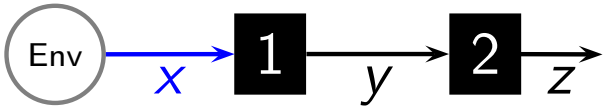
Realizability $\neq$ Satisfiability

reason: incomplete information
process 2 does not know x.

decisions of process 2
may not depend on x.



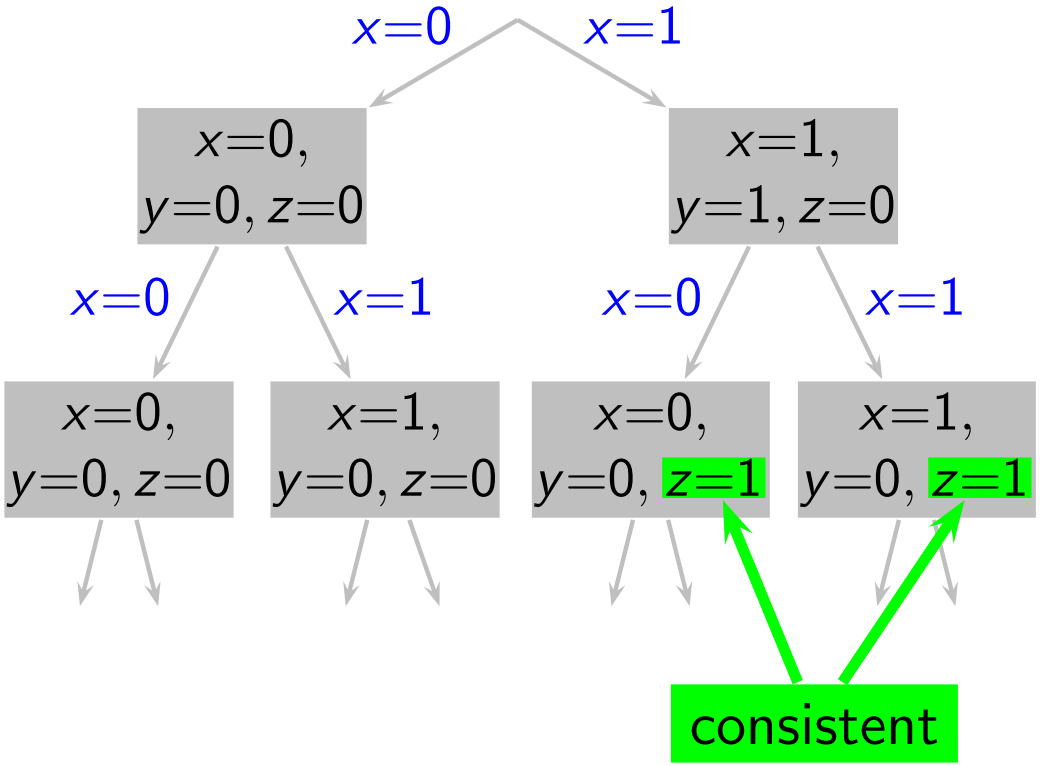
Synthesis of Multi-Process Systems



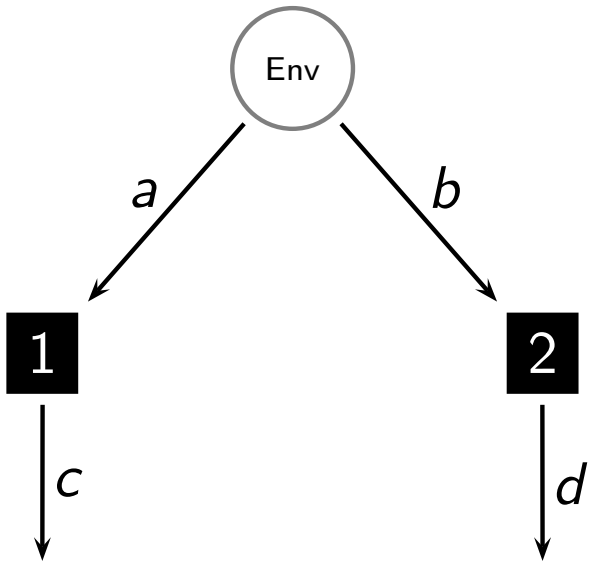
Realizability $\neq$ Satisfiability

reason: incomplete information
process 2 does not know x.

decisions of process 2
may not depend on x.



Distributed Synthesis is Undecidable



Pnueli, Rosner 1990

In general, the realizability question is undecidable for multi-process systems.

Information Forks

Information Fork

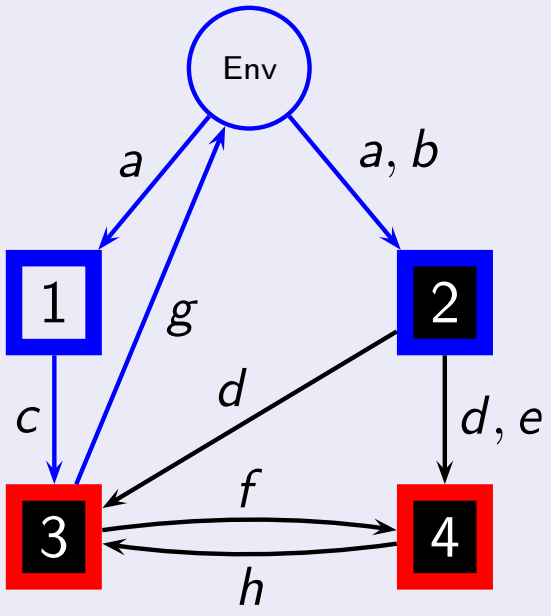
An information fork consists of a pair of black-box processes that are incomparable with respect to the informedness preorder $\preceq$:

$p \preceq q$: process p can see or derive all input to q.

Construction:

E_p: set of edges that carry data invisible to q

$p \preceq q \Leftrightarrow$ there is no directed path from Env to q in E_p



F., Schewe 2005

The realizability question is undecidable iff the architecture contains an information fork.

Information Forks

Information Fork

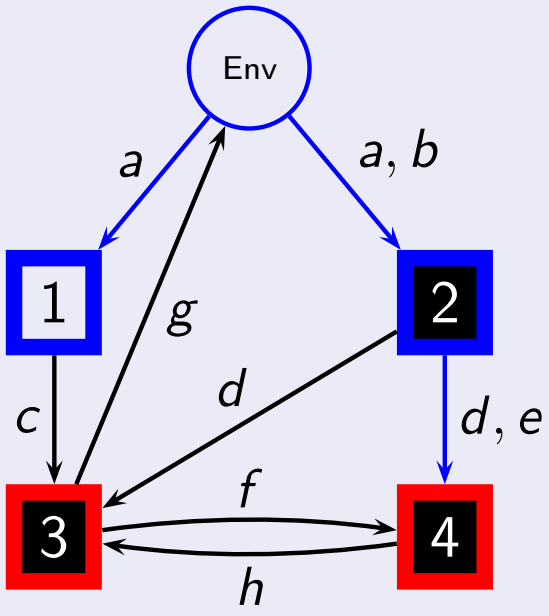
An information fork consists of a pair of black-box processes that are incomparable with respect to the informedness preorder $\preceq$:

$p \preceq q$: process p can see or derive all input to q.

Construction:

E_p: set of edges that carry data invisible to q

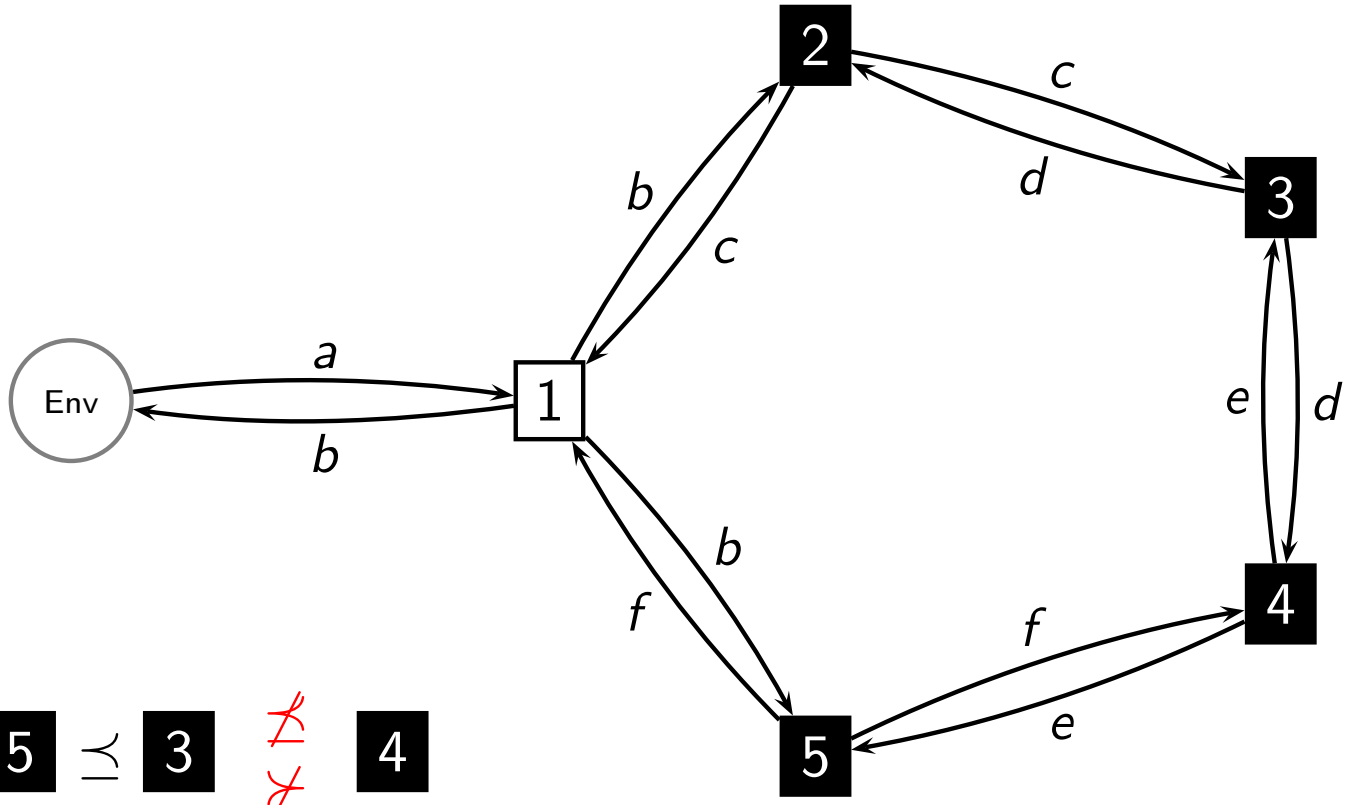
$p \preceq q \Leftrightarrow$ there is no directed path from Env to q in E_p



F., Schewe 2005

The realizability question is undecidable iff the architecture contains an information fork.

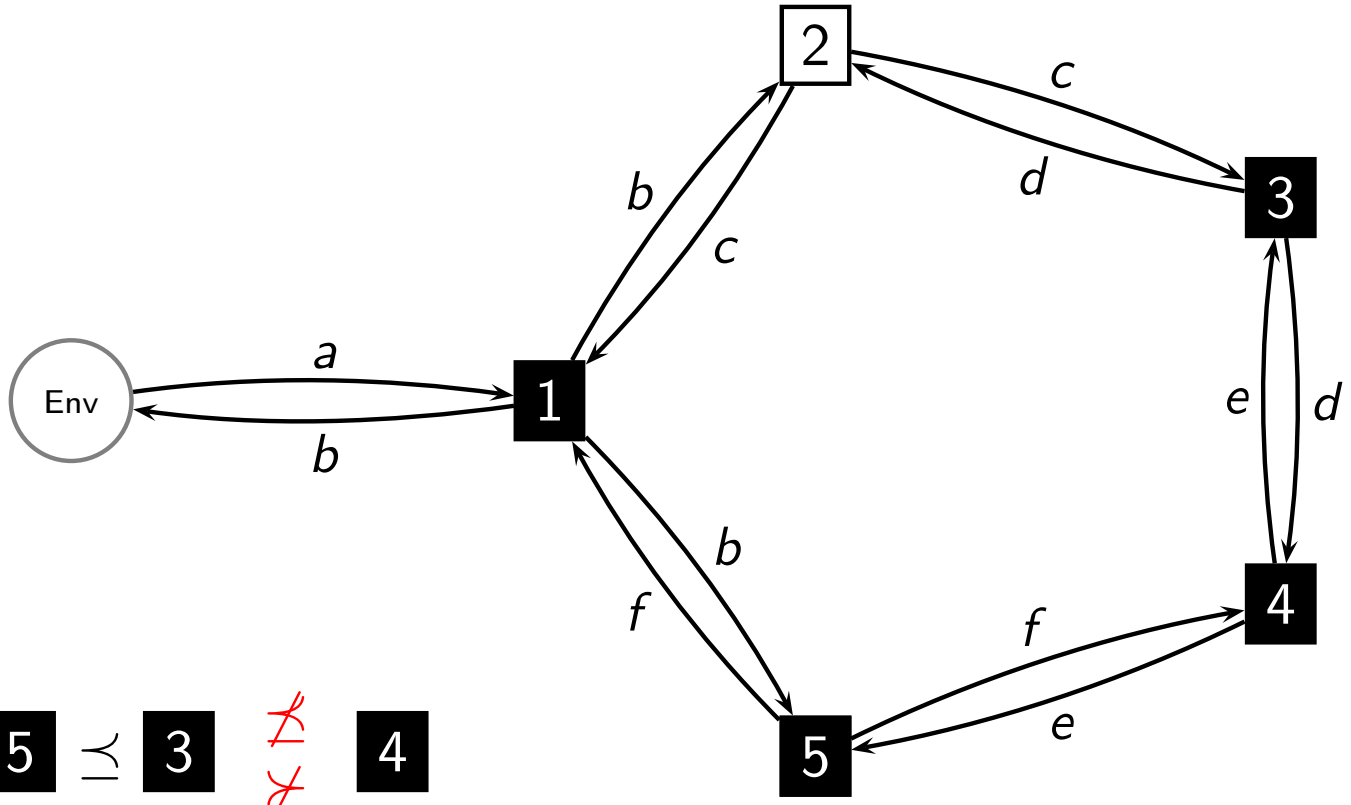
Example



$2 \simeq 5 \preceq 3 \not\preceq \not\succeq 4$

information fork

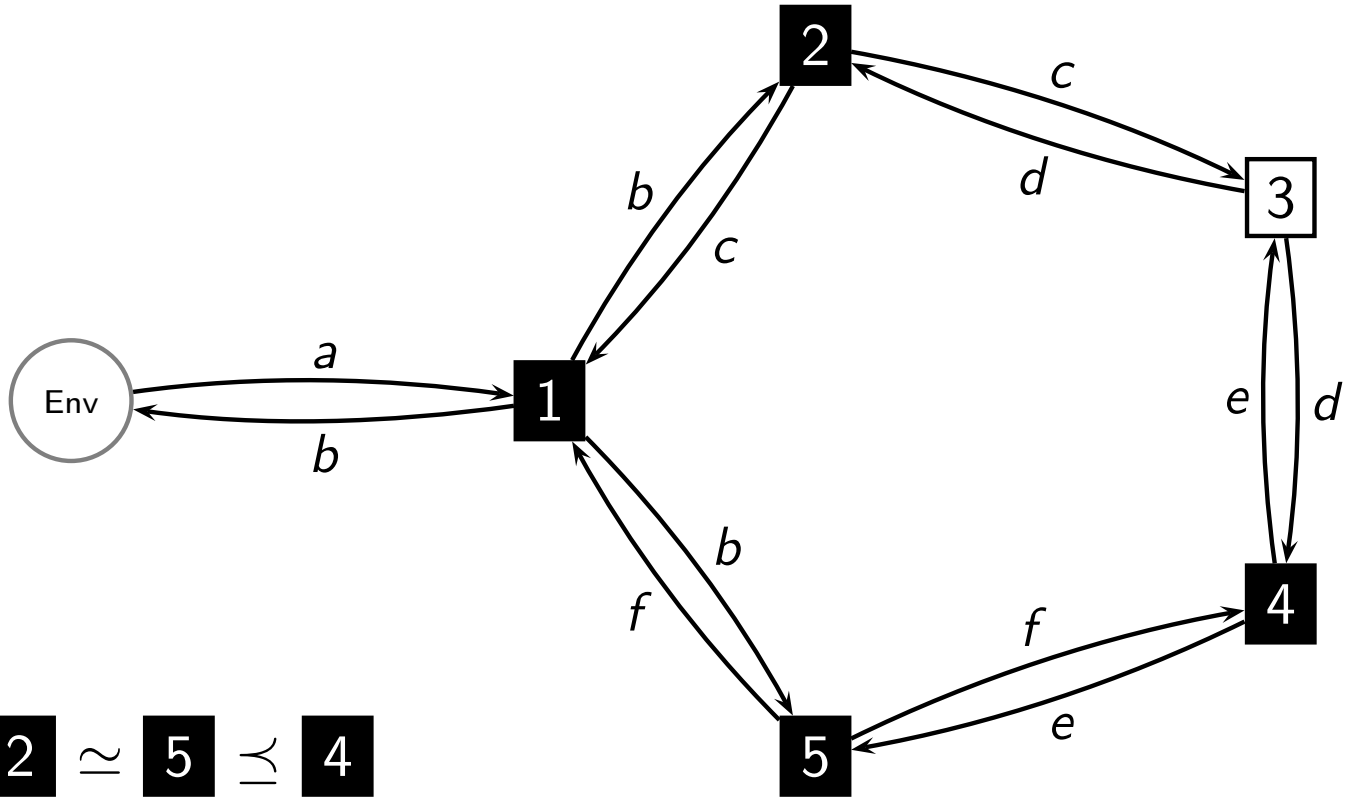
Example



$1 \preceq 5 \preceq 3 \not\preceq \not\succeq 4$

information fork

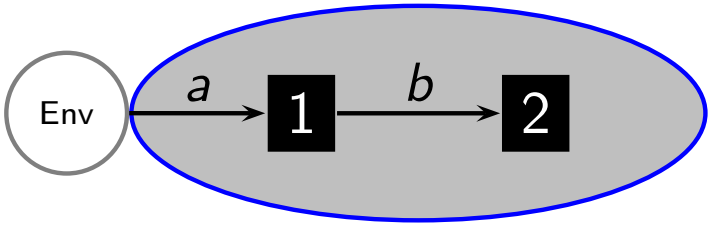
Example



$1 \preceq 2 \simeq 5 \preceq 4$

fork-free

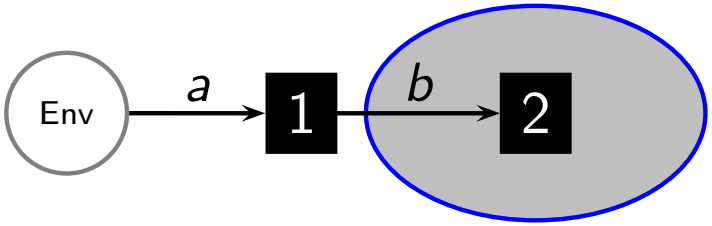
Automata-theoretic Synthesis: Pipelines



- $\mathcal{A}_\varphi$ — accepts implementations for super-process.

$|\mathcal{A}_\varphi| \in 2\mathrm{EXP}(|\varphi|)$.

Automata-theoretic Synthesis: Pipelines



- $\mathcal{A}_\varphi$ — accepts implementations for super-process.

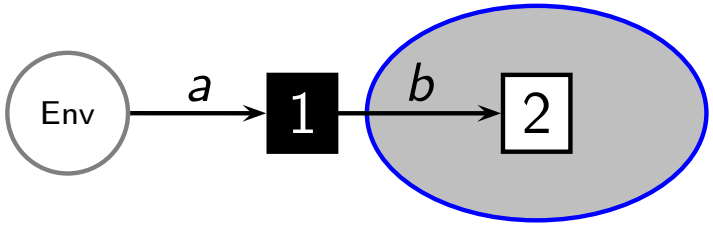
  $|\mathcal{A}_\varphi| \in 2\mathrm{EXP}(|\varphi|)$.

- $\mathcal{B}_\varphi$ — accepts an implementation for 2 iff

  there is an implementation for process 1 *such that their composition is accepted by* $\mathcal{A}_\varphi$

  $|\mathcal{B}_\varphi| \in \mathrm{EXP}(|\mathcal{A}_\varphi|)$.

Automata-theoretic Synthesis: Pipelines



- $\mathcal{A}_\varphi$ — accepts implementations for super-process.

  $|\mathcal{A}_\varphi| \in 2\mathrm{EXP}(|\varphi|)$.

- $\mathcal{B}_\varphi$ — accepts an implementation for $\boxed{2}$ iff

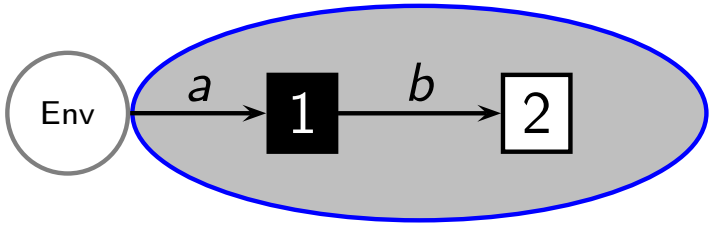
  there is an implementation for process $\boxed{1}$ *such that their composition is accepted by* $\mathcal{A}_\varphi$

  $|\mathcal{B}_\varphi| \in \mathrm{EXP}(|\mathcal{A}_\varphi|)$.

- If $\mathcal{B}_\varphi$ is nonempty it accepts an implementation $\boxed{2}$

  with $\boxed{2} \in 3\mathrm{EXP}(|\varphi|)$.

Automata-theoretic Synthesis: Pipelines



- $\mathcal{A}_\varphi$ — accepts implementations for super-process.

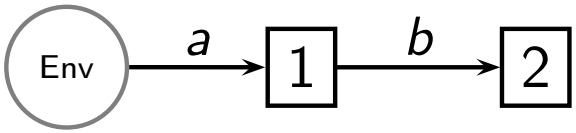
  $|\mathcal{A}_\varphi| \in 2\mathrm{EXP}(|\varphi|)$.
- $\mathcal{B}_\varphi$ — accepts an implementation for $\boxed{2}$ iff

  there is an implementation for process $\boxed{1}$ *such that their composition is accepted by* $\mathcal{A}_\varphi$

  $|\mathcal{B}_\varphi| \in \mathrm{EXP}(|\mathcal{A}_\varphi|)$.
- If $\mathcal{B}_\varphi$ is nonempty it accepts an implementation $\boxed{2}$

  with $\boxed{2} \in 3\mathrm{EXP}(|\varphi|)$.
- $\mathcal{A}'_\varphi$ — accepts proper implementations for $\boxed{1}$.

Automata-theoretic Synthesis: Pipelines



- $\mathcal{A}_\varphi$ — accepts implementations for super-process.

  $|\mathcal{A}_\varphi| \in 2\mathrm{EXP}(|\varphi|)$.
- $\mathcal{B}_\varphi$ — accepts an implementation for $\boxed{2}$ iff

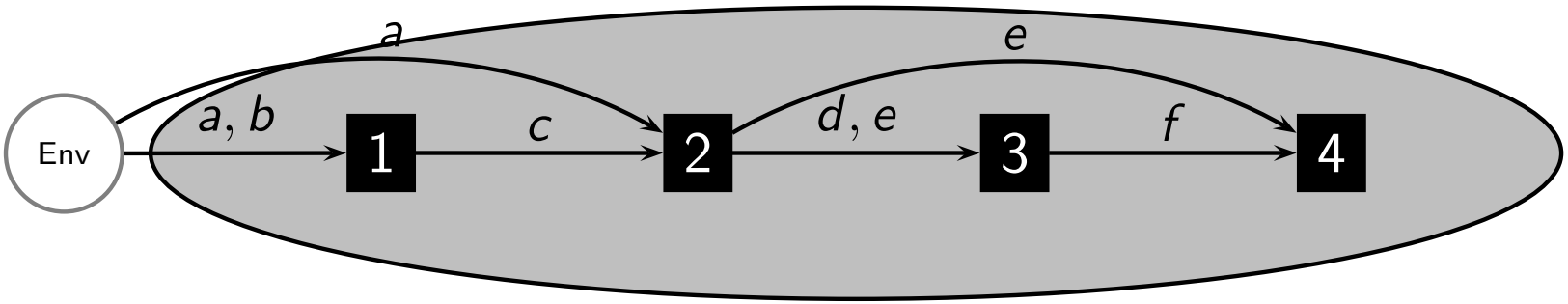
  there is an implementation for process $\boxed{1}$ *such that their composition is accepted by* $\mathcal{A}_\varphi$

  $|\mathcal{B}_\varphi| \in \mathrm{EXP}(|\mathcal{A}_\varphi|)$.
- If $\mathcal{B}_\varphi$ is nonempty it accepts an implementation $\boxed{2}$

  with $\boxed{2} \in 3\mathrm{EXP}(|\varphi|)$.
- $\mathcal{A}'_\varphi$ — accepts proper implementations for $\boxed{1}$.
- If $\mathcal{A}'_\varphi$ is non-empty, it accepts an implementation $\boxed{1}$

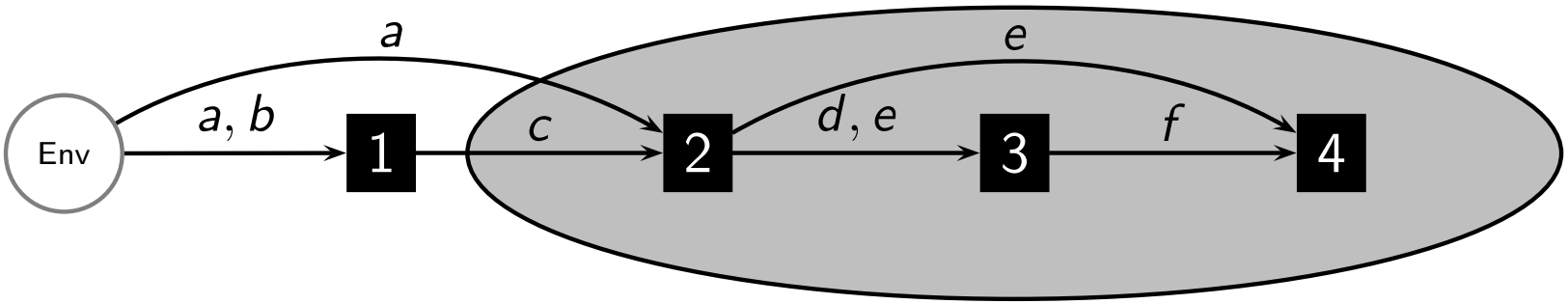
  with $\boxed{1} \in 3\mathrm{EXP}(|\varphi|)$.

Automata-theoretic Synthesis



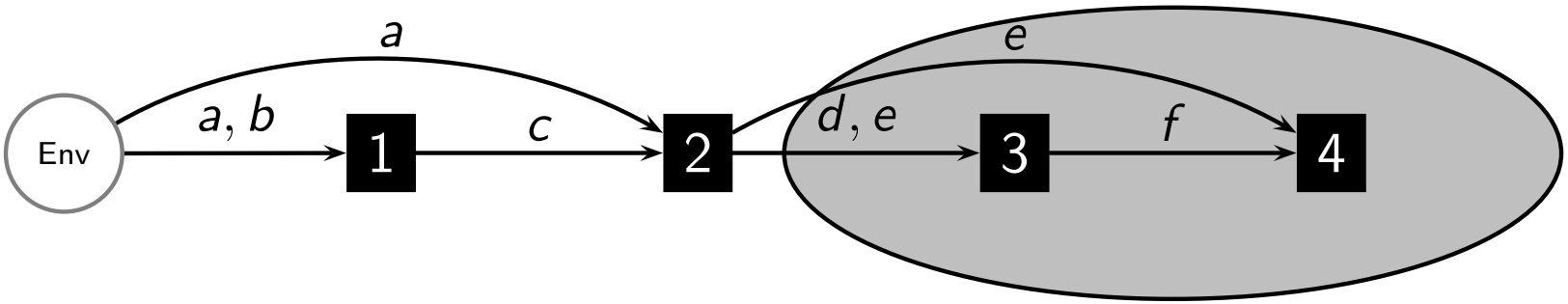
$$|\mathcal{A}_\varphi| \in 2\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



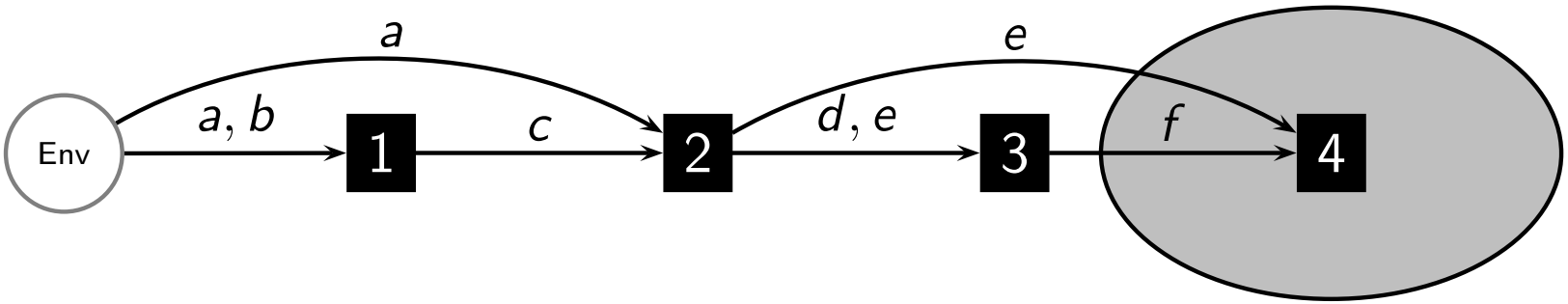
$$|\mathcal{B}_\varphi| \in 3\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



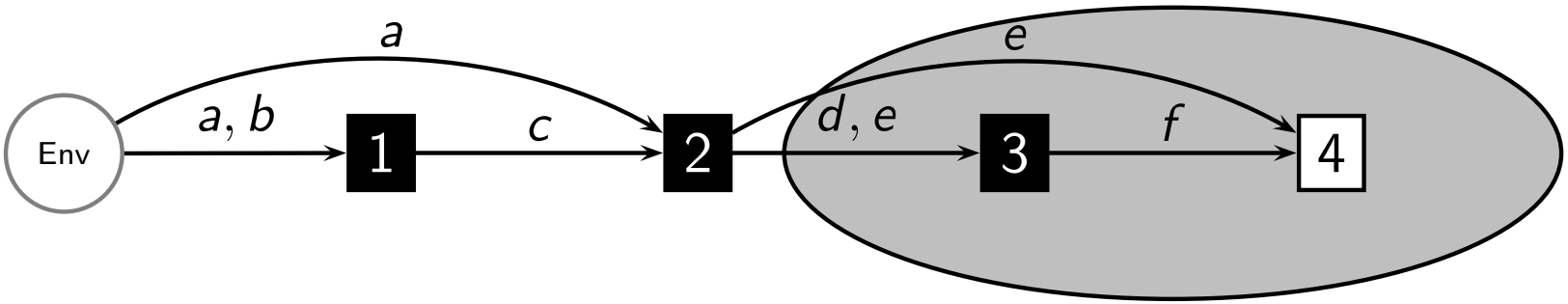
$$|C_\varphi| \in 4\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



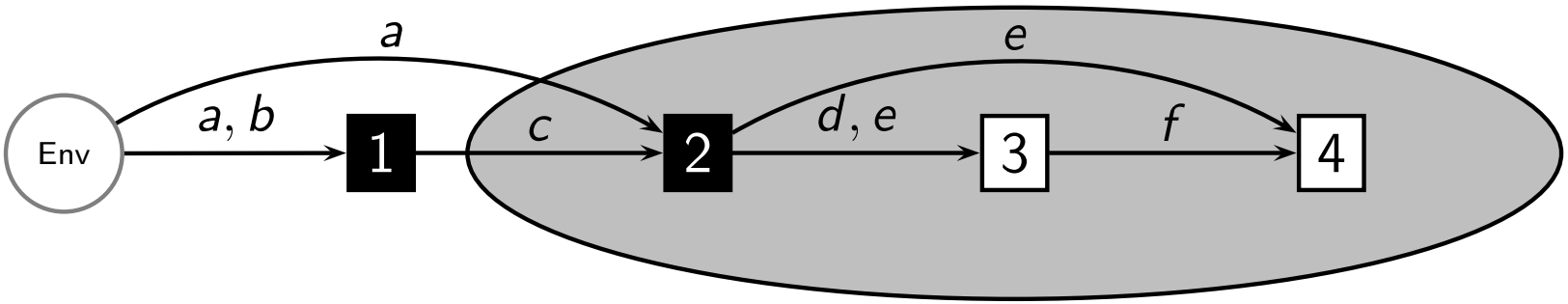
$$|\mathcal{D}_\varphi| \in 5\mathrm{EXP}(|\varphi|) \quad - \quad \boxed{4} \in 5\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



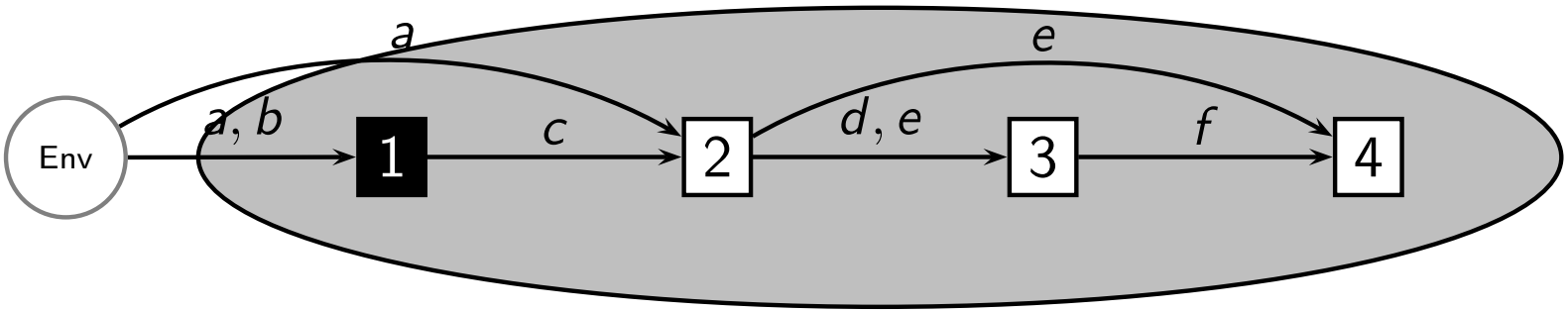
$$|\mathcal{C}'_\varphi| \in 5\mathrm{EXP}(|\varphi|) \quad - \quad \boxed{3} \in 5\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



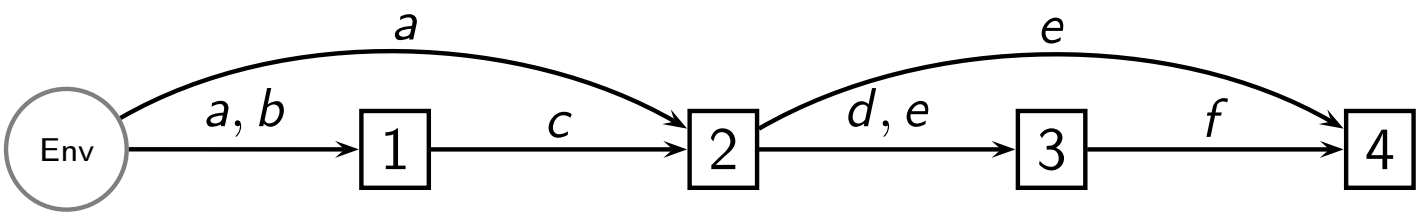
$$|\mathcal{B}'_\varphi| \in 5\mathrm{EXP}(|\varphi|) \quad - \quad \boxed{2} \in 5\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



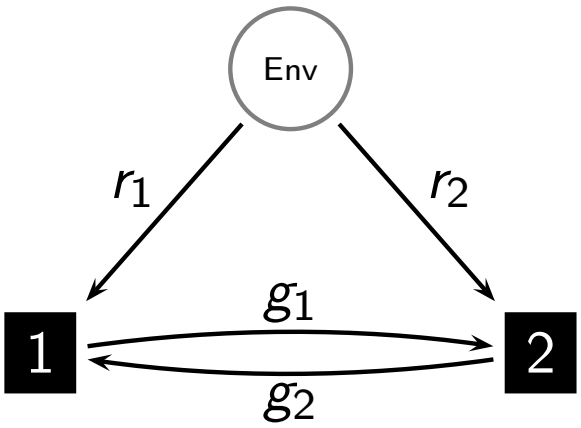
$$|\mathcal{A}'_\varphi| \in 5\mathrm{EXP}(|\varphi|) \quad - \quad \boxed{1} \in 5\mathrm{EXP}(|\varphi|)$$

Automata-theoretic Synthesis



$\boxed{1}, \boxed{2}, \boxed{3}, \boxed{4} \in 5\mathrm{EXP}(|\varphi|)$

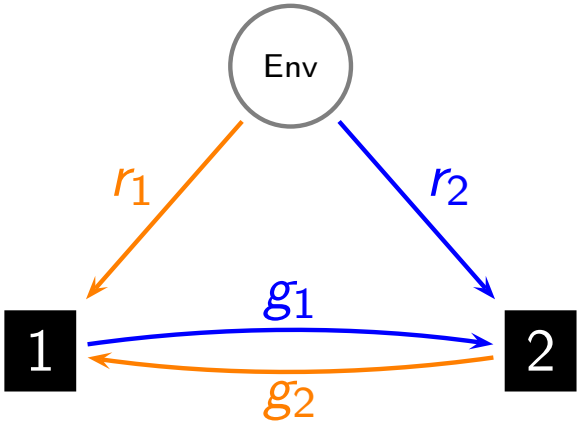
SMT-based Approach: Extended Constraint System



Local transition system

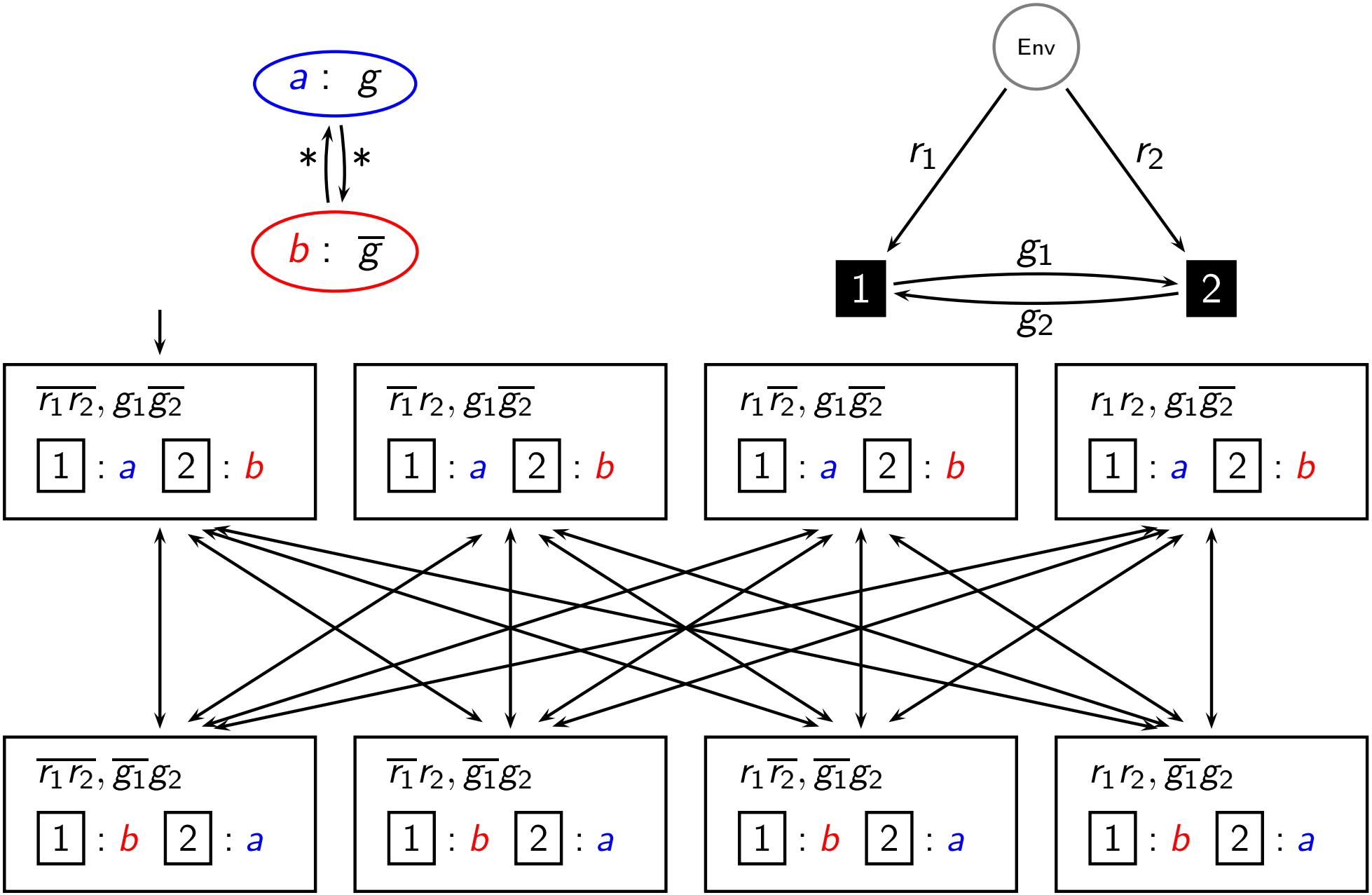
- projection from global states to local states for process i: $proj_i : \mathbb{N}_N \rightarrow \mathbb{N}_{N_i}$
- local transition function for local input and local state $\tau_{i;inp} : \mathbb{N}_{N_i} \rightarrow \mathbb{N}_{N_i}$

SMT-based Approach: Locality Constraint



- $\forall t.\, \tau_{1;r_1,g_2(proj_2(t))}\big(proj_1(t)\big) = proj_1(\tau_{r_1 r_2}(t)) = proj_1(\tau_{r_1 \overline{r}_2}(t))$
  $\wedge\ \tau_{1;\overline{r_1},g_2(proj_2(t))}\big(proj_1(t)\big) = proj_1(\tau_{\overline{r}_1 r_2}(t)) = proj_1(\tau_{\overline{r}_1 \overline{r}_2}(t))$
- $\forall t.\, \tau_{2;r_2,g_1(proj_1(t))}\big(proj_2(t)\big) = proj_2(\tau_{r_1 r_2}(t)) = proj_2(\tau_{\overline{r}_1 r_2}(t))$
  $\wedge\ \tau_{2;\overline{r_2},g_1(proj_1(t))}\big(proj_2(t)\big) = proj_2(\tau_{r_1 \overline{r}_2}(t)) = proj_2(\tau_{\overline{r}_1 \overline{r}_2}(t))$

Distributed Implementations



Experimental Results

Fully-informed arbiter

bound	4	5	6	7	8
result	unsatisfiable	satisfiable	satisfiable	satisfiable	satisfiable
# decisions	17566	30011	52140	123932	161570
# conflicts	458	800	1375	2614	3987
# boolean variables	1850	2854	3734	5406	6319
memory (MB)	18.3008	20.0586	22.5781	27.5000	35.7148
time (seconds)	0.21	0.63	1.72	5.15	12.38

(Using Yices on a 2.6 GHz Dual Core AMD Opteron)

Experimental Results

Distributed arbiter

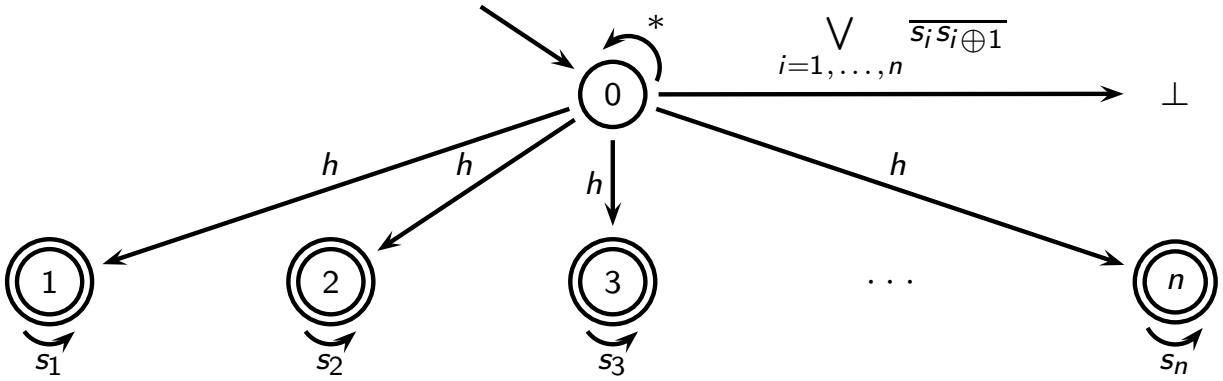
bound	4	5	6	7	8	9
result	unsat	unsat	unsat	unsat	sat	sat
# decisions	16725	47600	91480	216129	204062	344244
# conflicts	326	1422	8310	61010	11478	16347
# boolean variables	1890	7788	5793	13028	8330	10665
memory (MB)	18.0273	22.2109	28.5312	43.8594	42.2344	61.9727
time (seconds)	0.16	1.72	14.84	208.78	32.47	72.97

bound	8 (1)	8 (2)	8 (3)	8 (4)
result	unsat	unsat	sat	sat
# decisions	309700	1122755	167397	208255
# conflicts	92712	775573	13086	13153
# boolean variables	15395	25340	8240	7806
memory (MB)	54.1641	120.0160	42.1484	42.7188
time (seconds)	263.44	5537.68	31.12	30.36

Experimental Results

Dining philosophers

	3 states			4 states			6 states		
philosophers	time (s)	memory (MB)	result	time (s)	memory (MB)	result	time (s)	memory (MB)	result
125	1.52	23.2695	unsat	23.84	36.2305	unsat	236.5	87.7852	sat
250	5.41	29.2695	unsat	130.07	52.0859	sat	141.36	91.1328	sat
375	22.81	38.9727	unsat	128.83	58.1992	unsat	890.58	154.355	sat
500	17.98	39.9297	unsat	15.84	52.9336	sat	237.04	119.309	sat
625	35.57	49.5586	unsat	417.05	94.7188	unsat	486.5	130.977	sat
750	22.25	52.3359	unsat	20.85	69.1562	sat	82.63	99.707	sat
875	51.98	56.0859	unsat	628.84	119.363	unsat	2546.88	255.965	sat
1000	168.17	70.3906	unsat	734.74	117.703	sat	46.18	124.691	sat
1375	104.27	75.4531	unsat	3518.85	191.113	unsat	8486.18	490.566	sat
1750	169.93	97.543	unsat	107.14	126.477	sat	257.77	185.887	sat
2125	163.39	113.27	unsat	5932.24	315.711	unsat	6465.75	424.121	sat
2500	375.29	135.535	unsat	235.17	202.59	sat	319.78	253.781	sat
3000	666.18	155.57	unsat	533.23	228.961	sat	3158.26	493.617	sat
3500	308.23	169.348	unsat	897.6	270.676	sat	829.52	398.008	sat
5000	982.68	240.273	unsat	3603.7	421.832	sat	1357.48	582.457	sat
7000	2351.87	313.277	unsat	7069.55	535.98	sat	6438.73	1081.68	sat
10000	4338.83	448.648	unsat	4224.28	761.008	sat	10504.6	1121.58	sat



Conclusions

- SMT-based approach delivers small implementations fast.
  Is there a system of reasonable size?
- Automata-theoretic approach ensures decidability.
  But: Representing all possible implementations is too expensive.
- Challenge: Specialized solvers?