Effective Boolean Methods for Reasoning About Knowledge

Alessandro Cimatti

Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy

cimatti@irst.itc.it

Motivations

Reasoning about dynamic systems in order to...

- ... identify suitable courses of actions:
  Planning
- ... detect reason for system malfunctioning:
  Diagnosis
- .. detect if something might go be malfunctioning and go undetected:
  Diagnosability

Key challenge:

- reasoning agents must deal with incomplete information...
- ... induced by unobservable, hidden states

In this talk

Effective methods for reasoning with/about uncertainty/partial knowledge.

Focus on

- planning under partial observability using BDD's.
- plan formation rather than verification
- same basic ideas

Outline of the presentation:

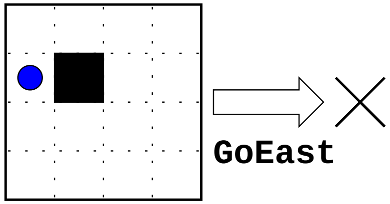
- reasoning about dynamic systems in in belief space
- conformant planning
- observations: planning under partial observability
- related problems/techniques
- conclusions

Some Motivating Examples

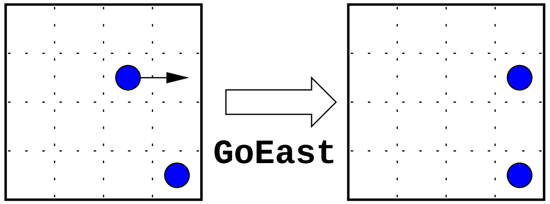
Nondeterministic Domains

We consider a general model of nondeterministic domains, where actions have

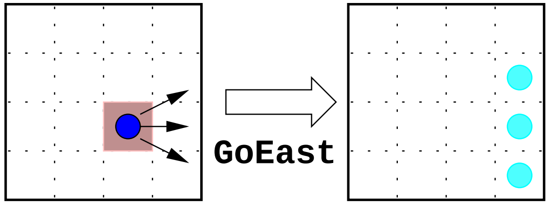
- Preconditions:



- Conditional effects:



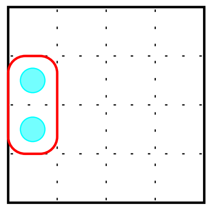
- Uncertain effects:



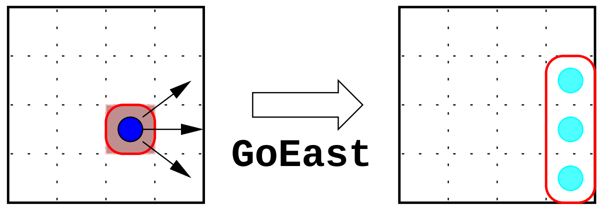
Representing Incomplete Information with Belief States

Incompleteness of information can be due to:

- Incompletely specified initial condition:



- Nondeterministic action effects:

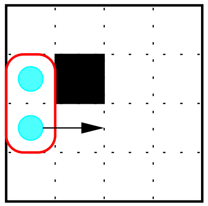


Indistinguishable states are collected into sets called Belief States.

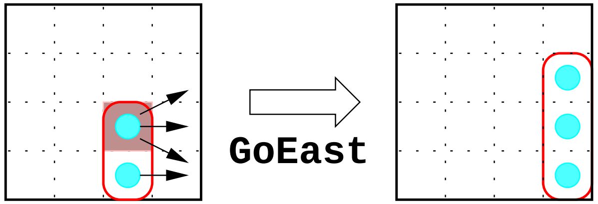
The Belief Space is the set of all belief states, i.e. $Pow(\mathcal{S})$.

Actions in Belief Space

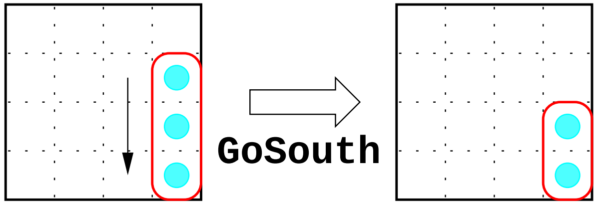
- Action applicability must be guaranteed



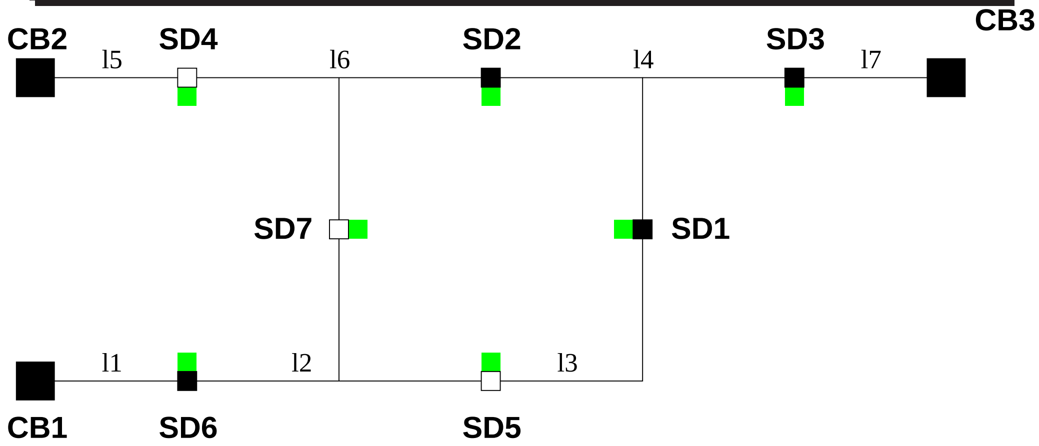
- Action execution gathers all effects



- Action execution can reduce uncertainty

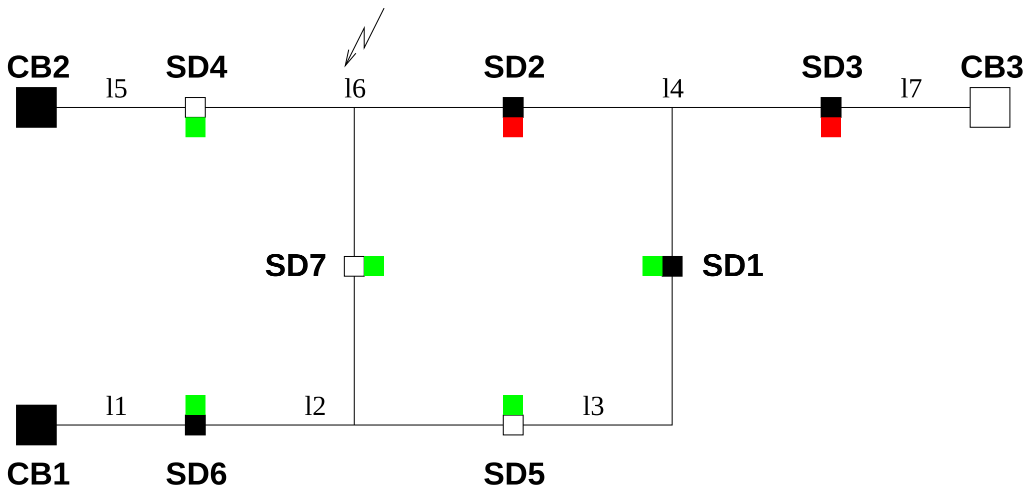


The Power Supply Restoration Domain



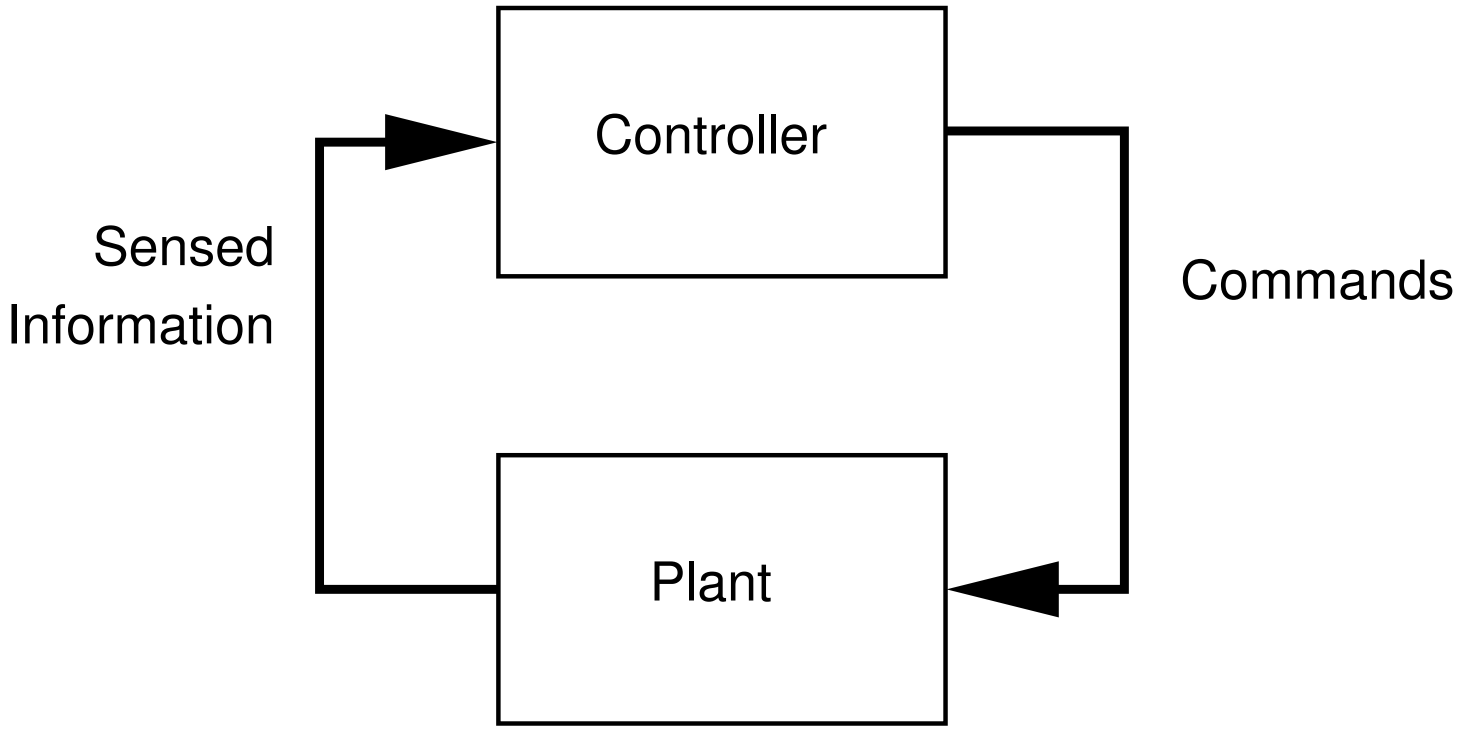
- Circuit Breakers (CB) provide power, Switching Devices (SD) connect lines
- Lines provide power to end users
- Power from different CB, depending on SD status
  - open: white square; closed: black square
- Faults on line (short) causes the feeding CB to automatically open: black out
- The PRS problem in a nutshell:
  - locate and isolate fault
  - reconfigure network to supply power wherever possible

Sensing the Network Status



- SD and CB equipped with sensors that can detect faults “downstream”
  - no fault downstream: green
  - fault downstream: red
- Can we locate the fault?
  - in this case, identification is easy: “downstream” to SD2
- How can we solve the problem?
  - open SD2

Planning under Partial Observability



Find a plan to be executed by the controller such that:

- it can be safely executed despite run-time uncertainty
- all the associated runs end in the goal

Planning in real world domains

Sources of incomplete information:

- Initial condition is incompletely specified;
- Action effects may be nondeterministic and unpredictable;
- Run-time observability can be limited.

Planning under Partial Observability:

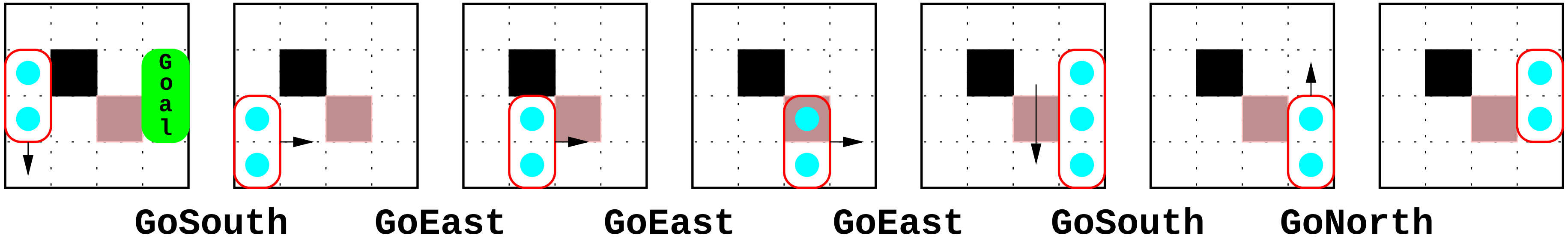
- build safely applicable plans, guaranteed to reach the goal
- from an incompletely specified initial condition
- despite nondeterministic action effects
- in the (sub)case of conformant planning:
  - no information can be acquired at run time (no observability)
  - sequential plans
- and in the full case:
  - limited information can be acquired at run time
  - conditional plans, branching on acquired information

Conformant Planning

Conformant Planning Problem:

- a set of goal states $\mathcal{G} \subseteq \mathcal{S}$
- a set of initial states $\mathcal{I} \subseteq \mathcal{S}$

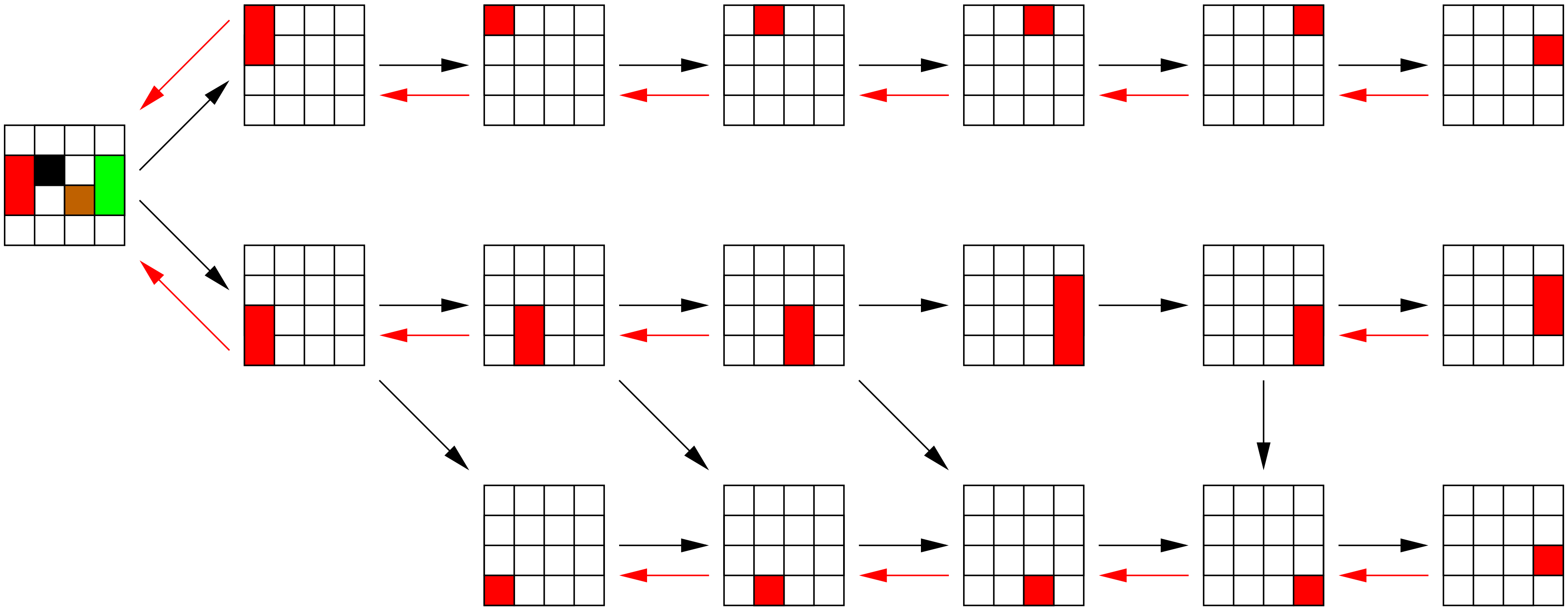
Example:



Conformant Plan:

- never violates action applicability;
- all resulting states are goal states.

The Search Space



Remark:

- the search space is $Pow(\mathcal{S})$
- $\mathcal{S}$ contains 15 states,
- $Pow(\mathcal{S})$ contains 32767 belief states!

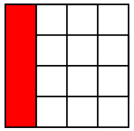
The Heuristic-Symbolic Approach to Conformant Planning

Key ingredients:

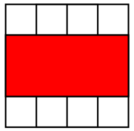
- Search formulated at level of Belief States
  - Belief state contains all information associated to plan execution
- BDD-based symbolic model checking techniques
  - Efficient storage of large state spaces
  - Expansion avoids transition enumeration
- Heuristic Search in Belief Space
  - Goal directed search

Belief States

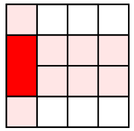
X = 0



Y in {1,2}

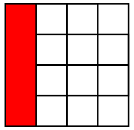


X = 0 and Y in {1,2}

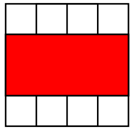


Belief States

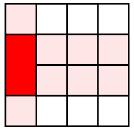
X = 0



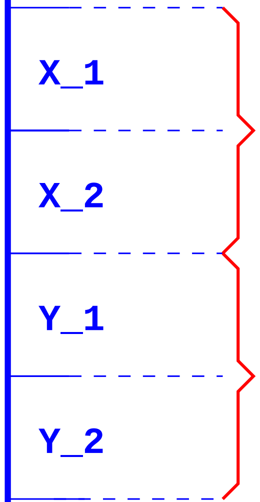
Y in {1,2}



X = 0 and Y in {1,2}



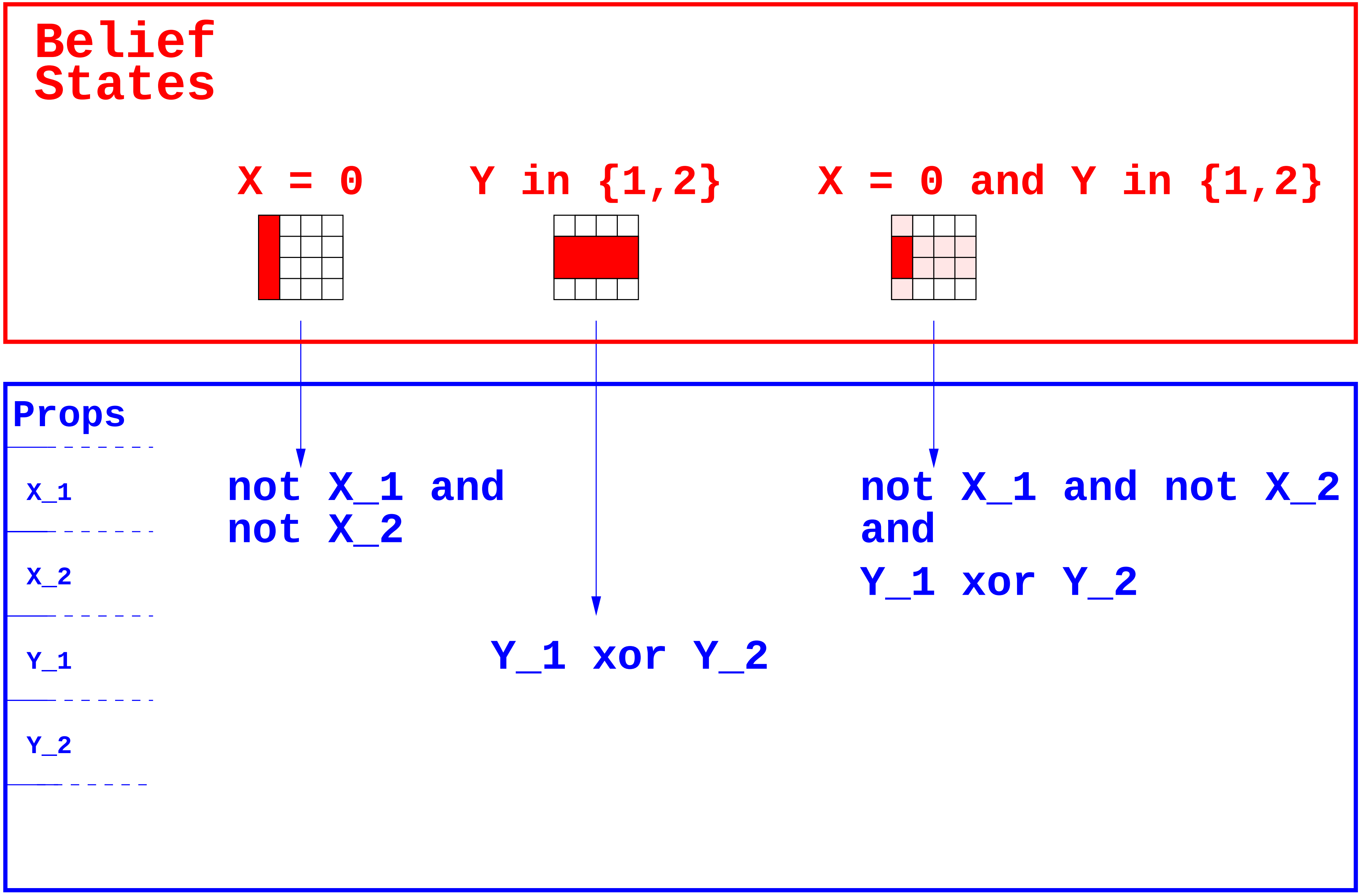
Props

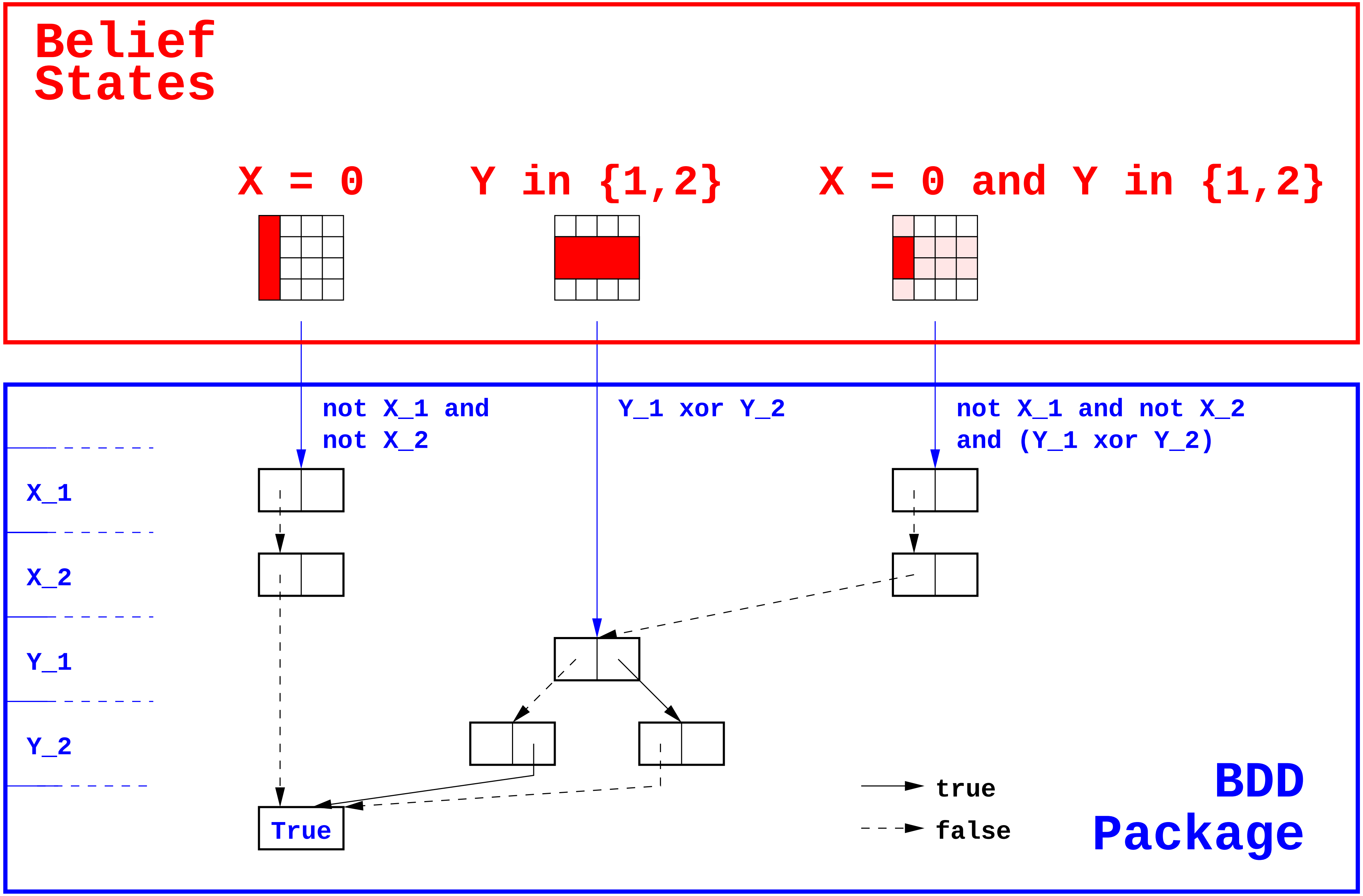


X:{0,1,2,3}

Y:{0,1,2,3}

X=0 → !X_1 & !X_2
X=1 → !X_1 & X_2
X=2 → X_1 & !X_2
X=3 → X_1 & X_2


Belief States
X = 0
Y in {1,2}
X = 0 and Y in {1,2}
Props
X_1
X_2
Y_1
Y_2
not X_1 and not X_2
Y_1 xor Y_2
not X_1 and not X_2 and
Y_1 xor Y_2

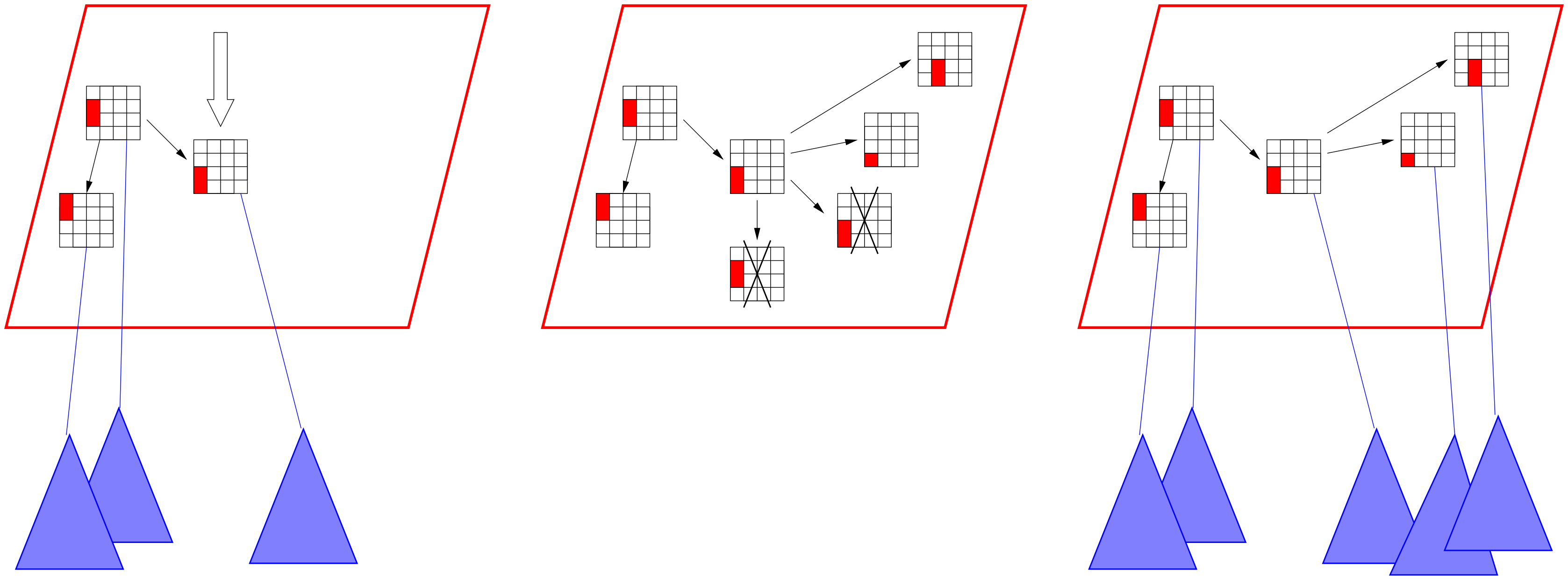

Belief States
X = 0
Y in {1,2}
X = 0 and Y in {1,2}
not X_1 and not X_2
Y_1 xor Y_2
not X_1 and not X_2 and (Y_1 xor Y_2)
X_1
X_2
Y_1
Y_2
True
true
false
BDD Package

BDD-based Search in Belief Space

Each belief state is associated to a BDD (unique thanks to canonical form).

Visited belief states hashed with BDD pointer used as key.

Search as iterated expansion of “best” belief states.



BDD-based Expansion of a Belief State

Key step is the efficient expansion of Bs, i.e. the computation of the set

$$\langle\langle\alpha_1 \;.\; \mathit{Exec}[\alpha_1](Bs)\rangle, \ldots, \langle\alpha_k \;.\; \mathit{Exec}[\alpha_k](Bs)\rangle\rangle$$

where α_i is applicable in Bs, and $\mathit{Exec}[\alpha_1](Bs)$ is new.

The effect of actions is represented by the BDD $\mathcal{R}(\mathbf{x}, \boldsymbol{\alpha}, \mathbf{x}')$, where $\mathbf{x}$, $\boldsymbol{\alpha}$ and $\mathbf{x}'$ are the current state, action and next state variables.

In terms of BDDs, the steps to compute the Expansion of Bs are the following:

- compute the BDD for all the actions applicable in Bs, i.e.

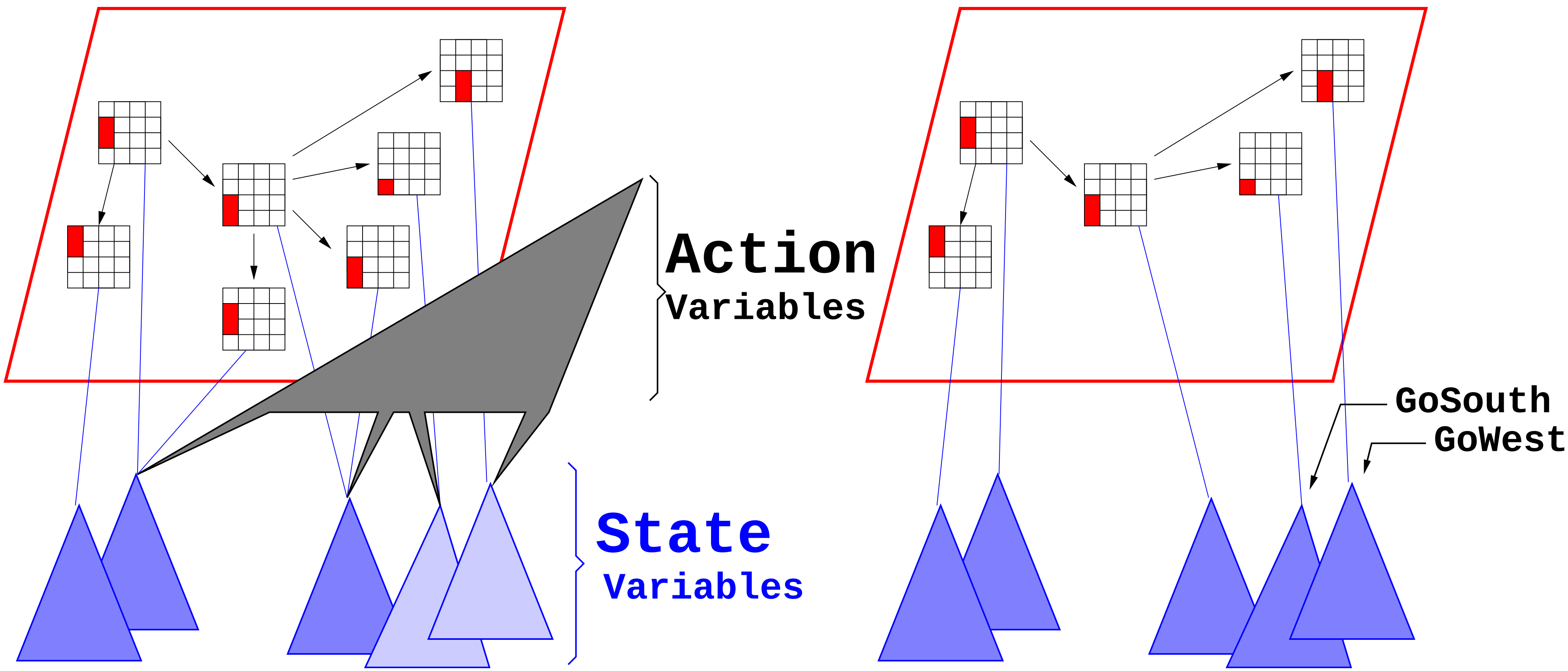
$$\textsc{ApplActs}_{Bs}(\boldsymbol{\alpha}) \doteq \forall \mathbf{x}.(Bs(\mathbf{x}) \rightarrow \exists \mathbf{x}'.\mathcal{R}(\mathbf{x}, \boldsymbol{\alpha}, \mathbf{x}'))$$

- compute the BDD

$$(\exists \mathbf{x}.(Bs(\mathbf{x}) \wedge \textsc{ApplActs}_{Bs}(\boldsymbol{\alpha}) \wedge \mathcal{R}(\mathbf{x}, \boldsymbol{\alpha}, \mathbf{x}')))[\mathbf{x}'/\mathbf{x}]$$

  that depends on $\mathbf{x}$ and $\boldsymbol{\alpha}$, and represents all the “applied action-possible result” pairs

Pruning the Expansion of a Belief State



Properties of Heursitic-Symbolic Conformant Planning

- Search can be performed either forward or backward.
- Termination: follows from finiteness of search space.
- If the problem is solvable, a conformant plan is returned.
- If the problem is not solvable, Failure is returned
  - cfr. bounded depth conformant planning
- Fully amenable to the use of heuristic functions.
  - (degree of uncertainty) cardinality of belief state
  - distance from goal condition

The role of Knowledge in Search Control

- Basic idea: heuristics based on distance between goal and “farthest” state in belief
- not enough!!!

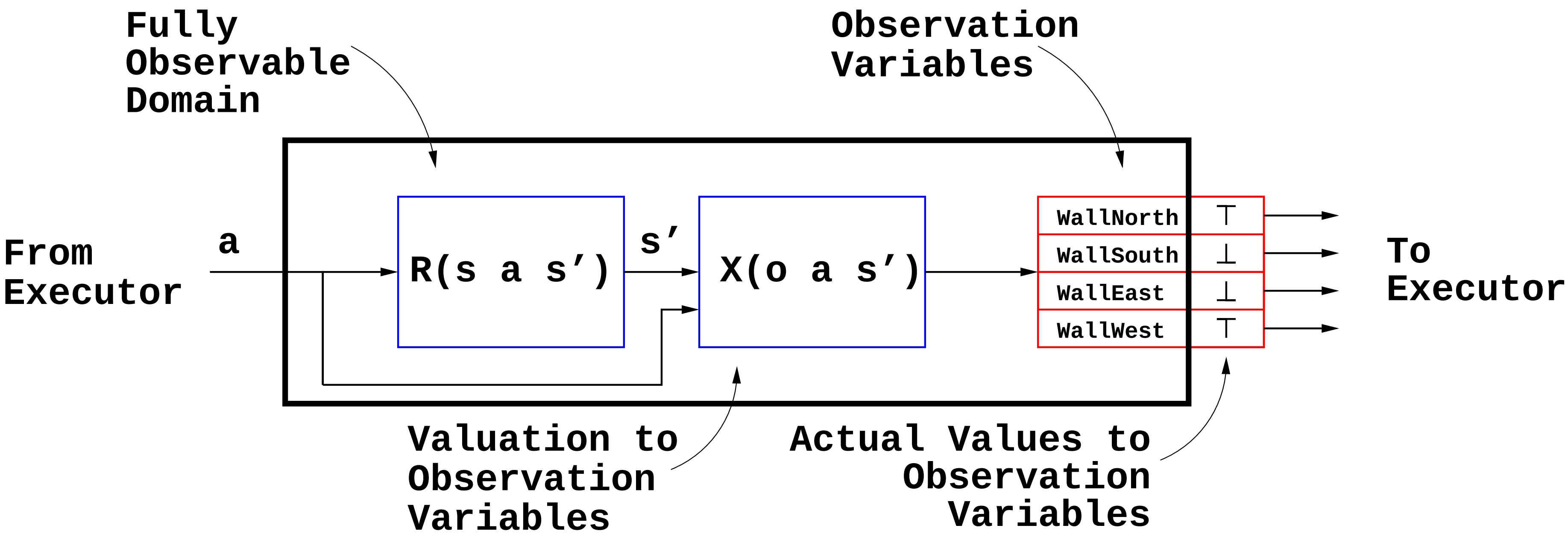
0	7	6	5	4	3	3	4	5
	6	5	4	3	2	2	3	4
	5	4	3	2	1	1	2	3
	4	3	2	1	0	0	1	2
	5	4	3	2	1	1	2	3
	6	5	4	3	2	2	3	4
	7	6	5	4	3	3	4	5
7	8	7	6	5	4	4	5	6
	0							7

6	5	4	3	2	2	3	4	0
5	4	3	2	1	1	2	3	
4	3	2	1	0	0	1	2	
4	3	2	1	0	0	1	2	
5	4	3	2	1	1	2	3	
6	5	4	3	2	2	3	4	
7	6	5	4	3	3	4	5	
8	7	6	5	4	4	5	6	7
0							7	

- must discover that $\mathcal{K}(\mathrm{x} = i)$ for some i is a necessary condition

The general case: partial observability

Modeling Partially Observable Nondeterministic Planning Domains



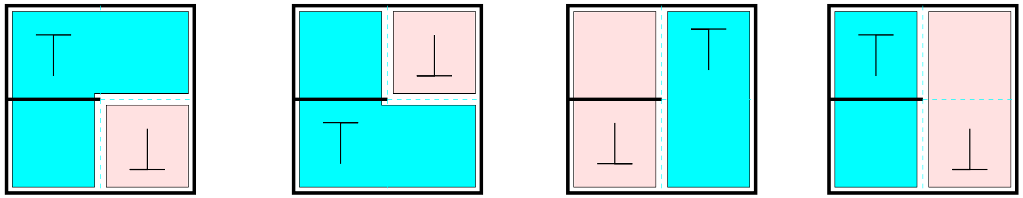
Remarks:

- Values to observation variables can depend on executed action, and on resulting state;
- Actions can affect the state of the domain and convey information at the same time;

Associating Values to Observation Variables: Examples

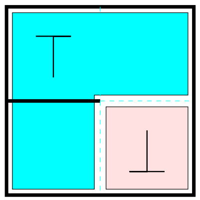
Action-independent observations:

`WallNorth` `WallSouth` `WallEast` `WallWest`



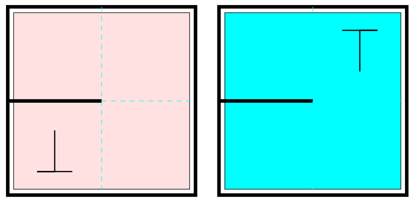
Action-dependent observations:

`WallNorth`



after `SenseWallNorth`

`WallNorth`

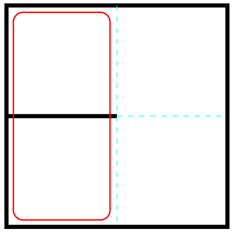


after a different action

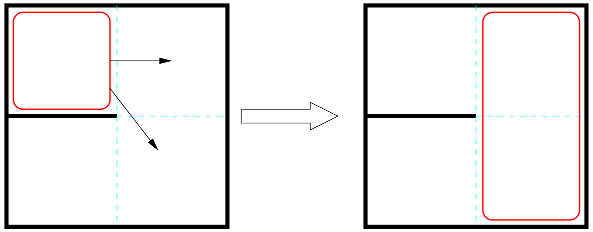
Representing Incomplete Information with Belief States

Reasons for incompleteness of information:

- Incompletely specified initial condition:



- Nondeterministic action effects:



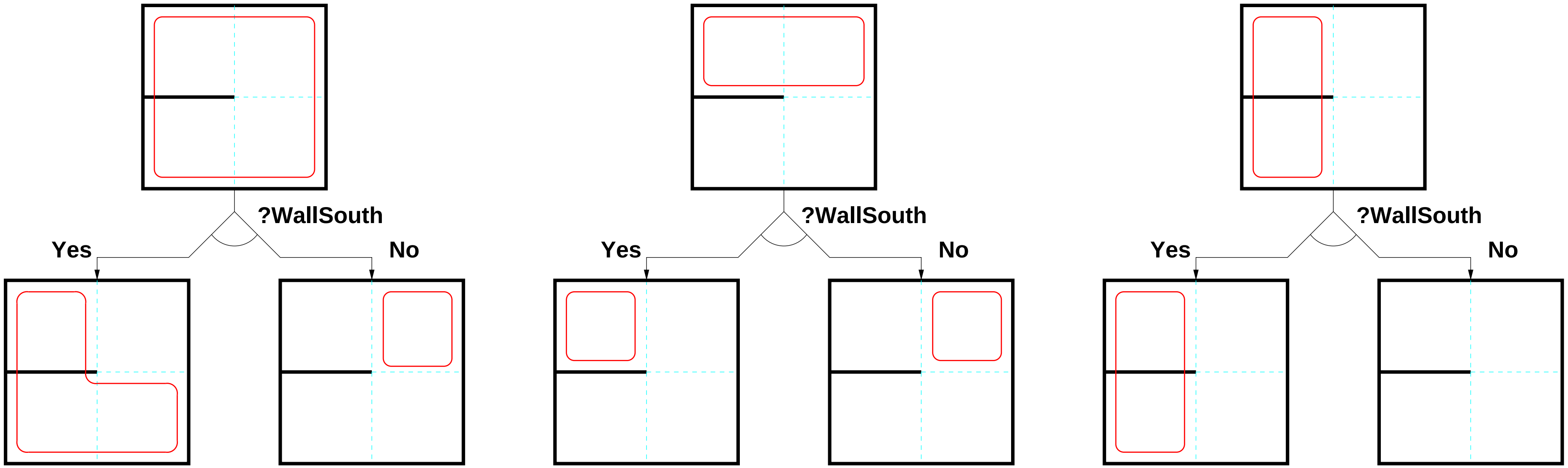
Indistinguishable states are collected into sets called Belief States.

The Belief Space is the set of all belief states, i.e. $Pow(\mathcal{S})$.

Actions are deterministic in Belief Space.

The role of Observations

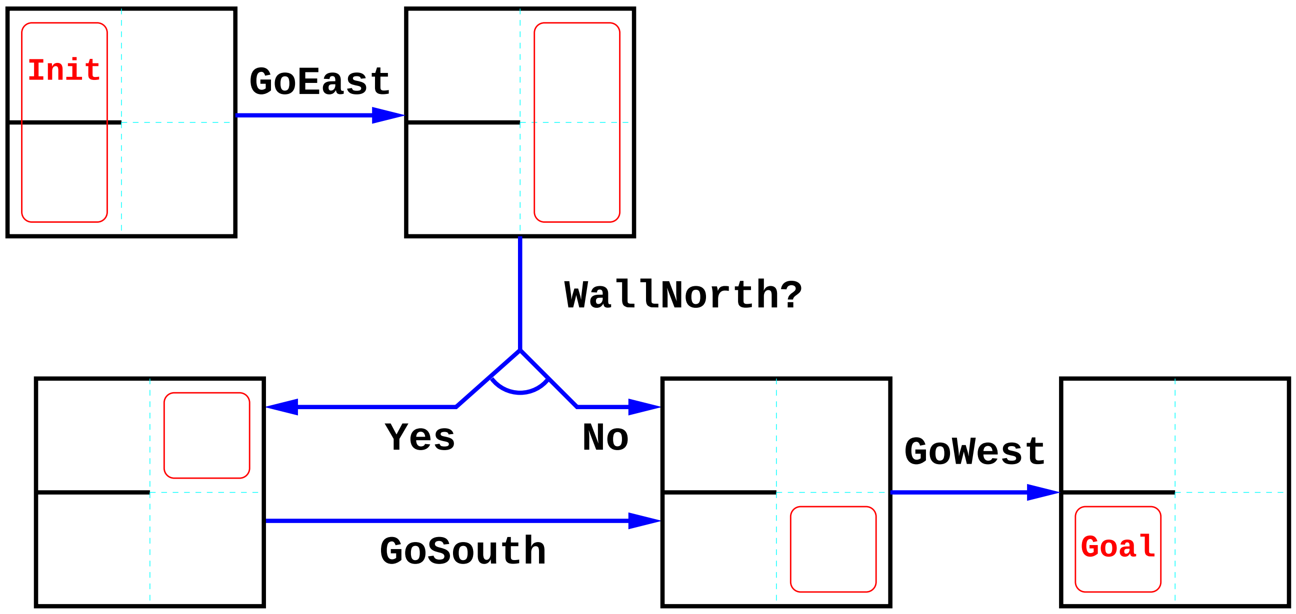
The value of observation variables may convey information and allow to “split” a Belief State.



Observations are nondeterministic in Belief Space.

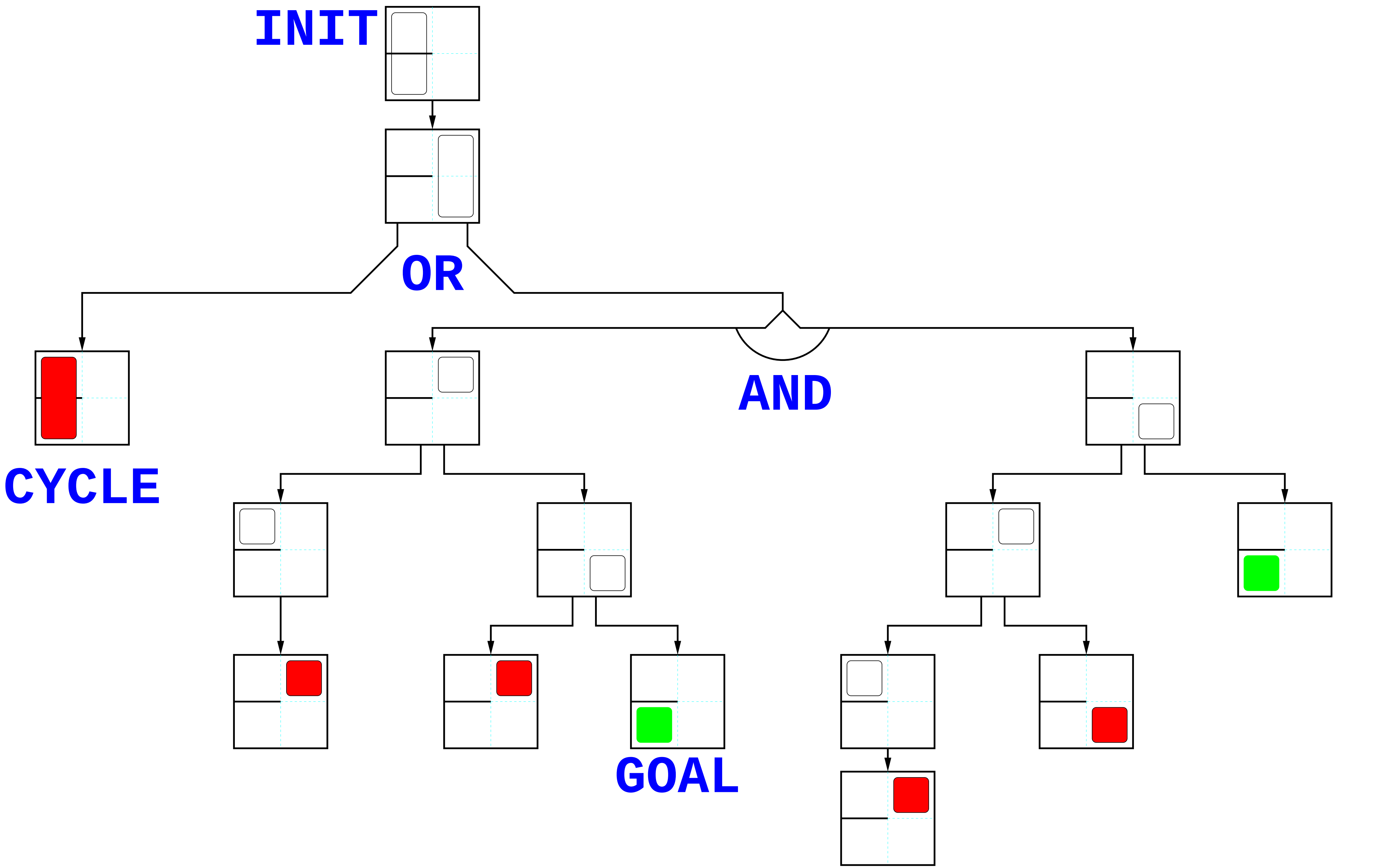
The Problem of Conditional Planning under Partial Observability

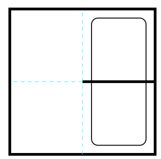
Conditional Planning Problem: $\mathcal{I} = \langle NW, SW \rangle$, $\mathcal{G} = \langle SW \rangle$.

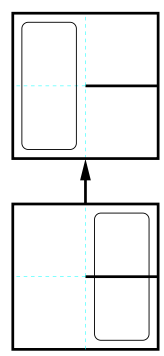


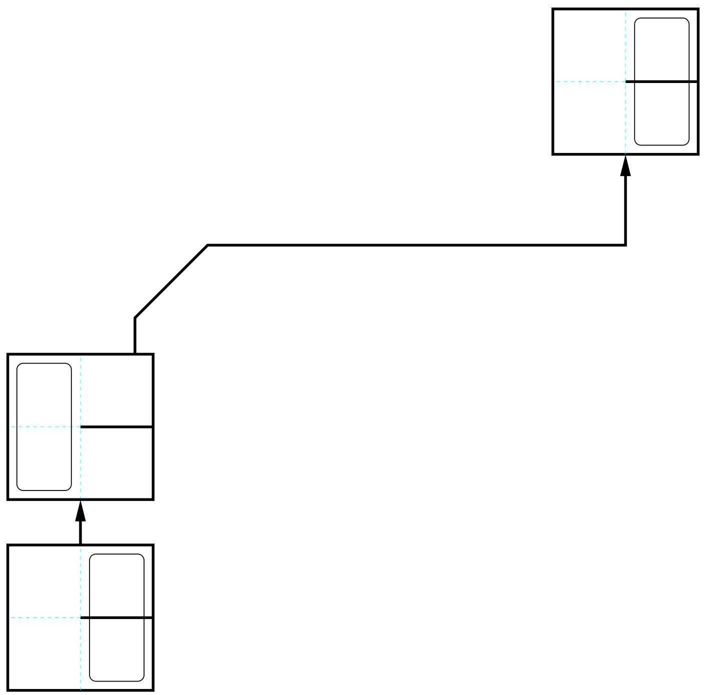
Solution plan: `GoEast`; **if** `WallNorth`**? then** (`GoSouth`; `GoWest`) **else** `GoWest`

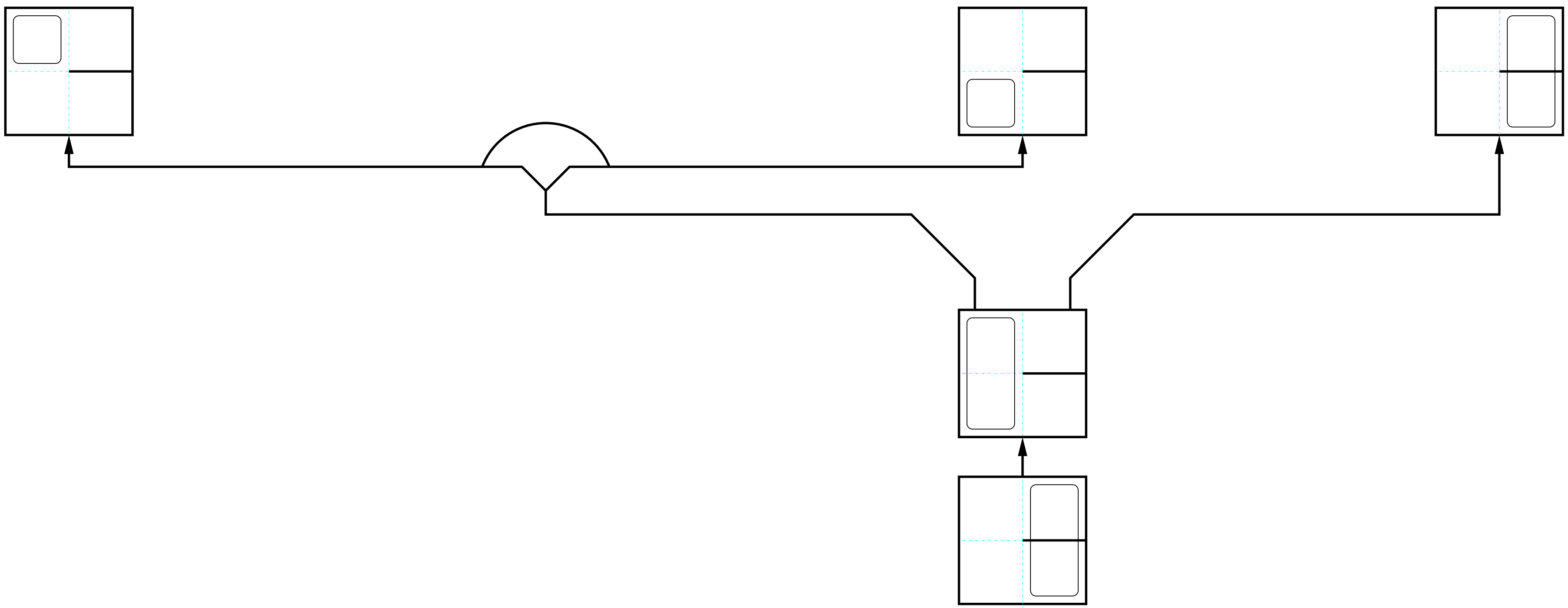
The search space

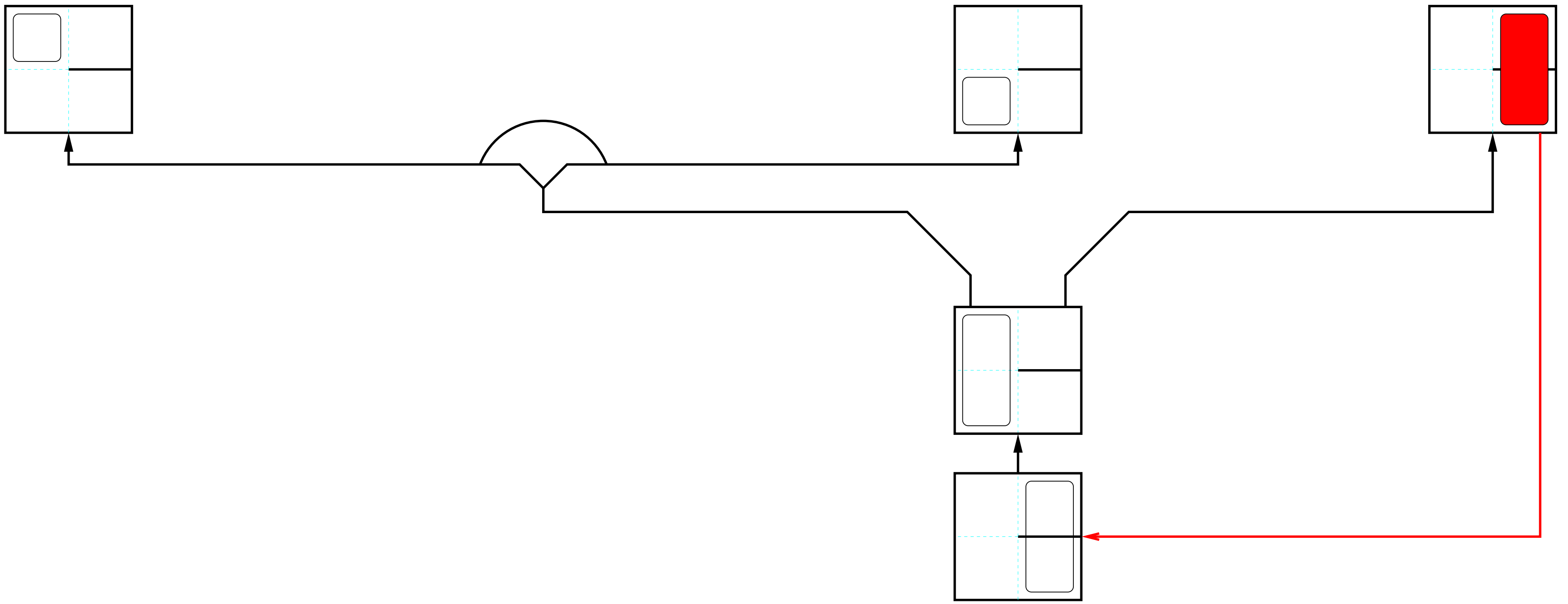


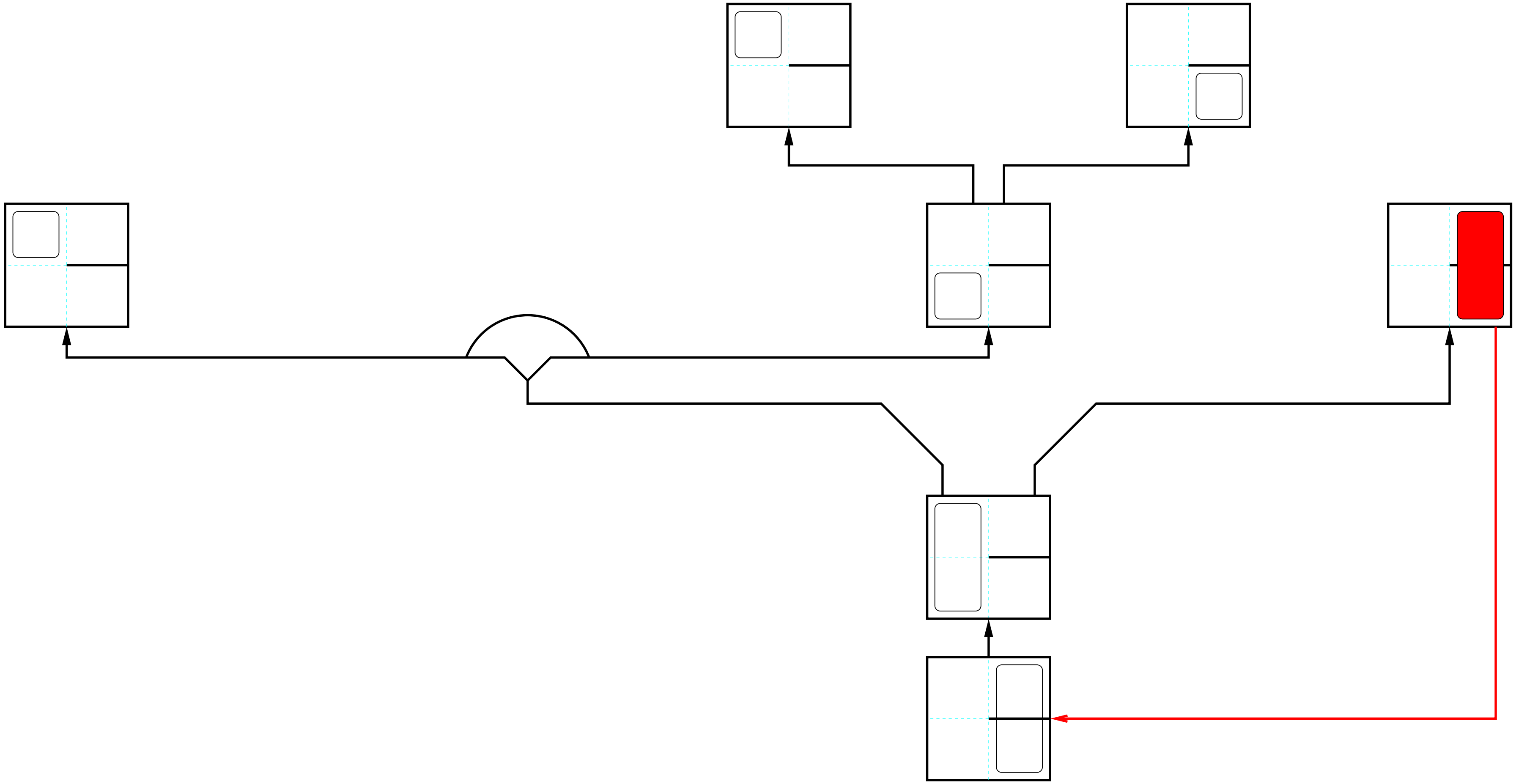


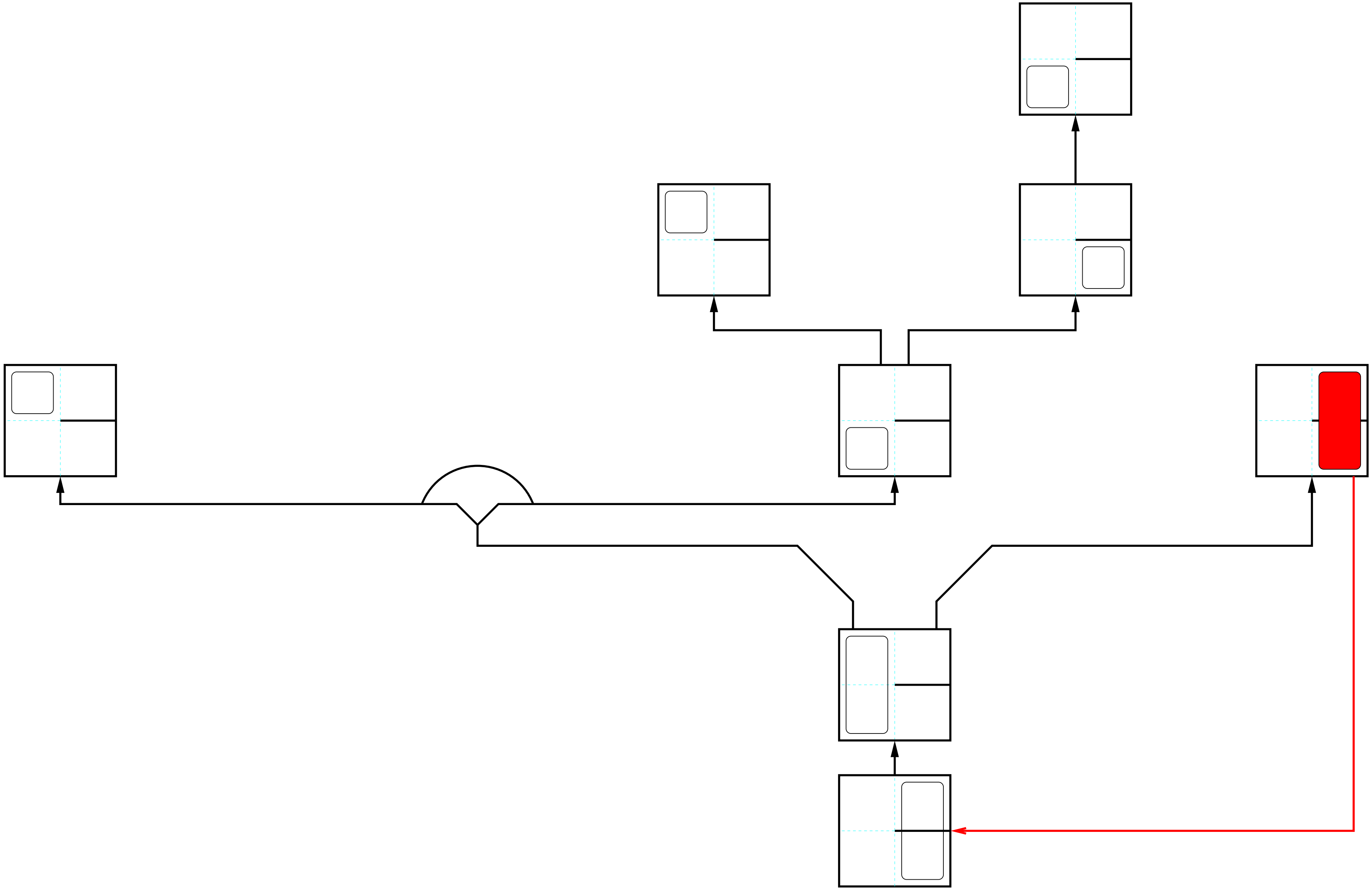


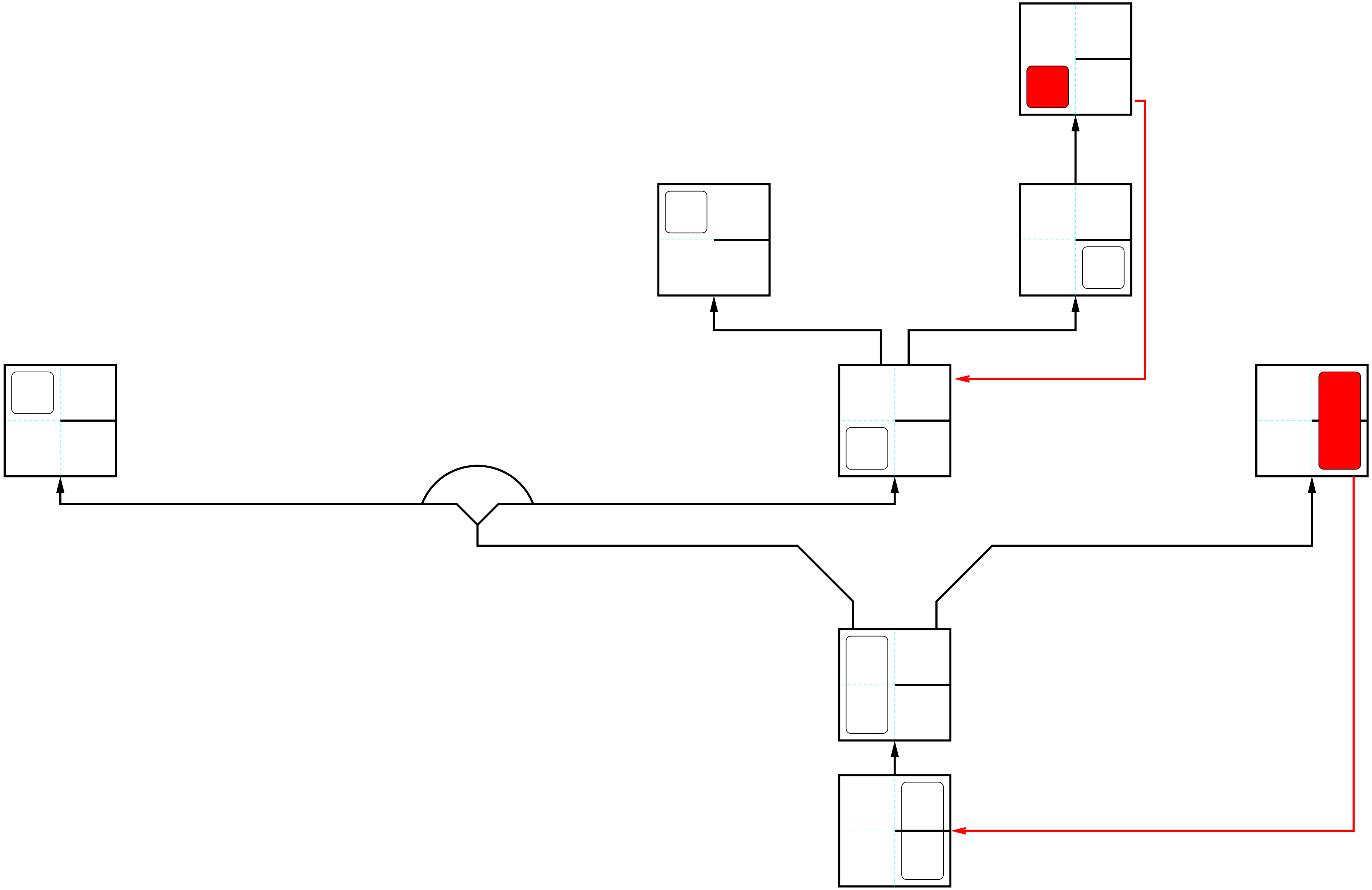


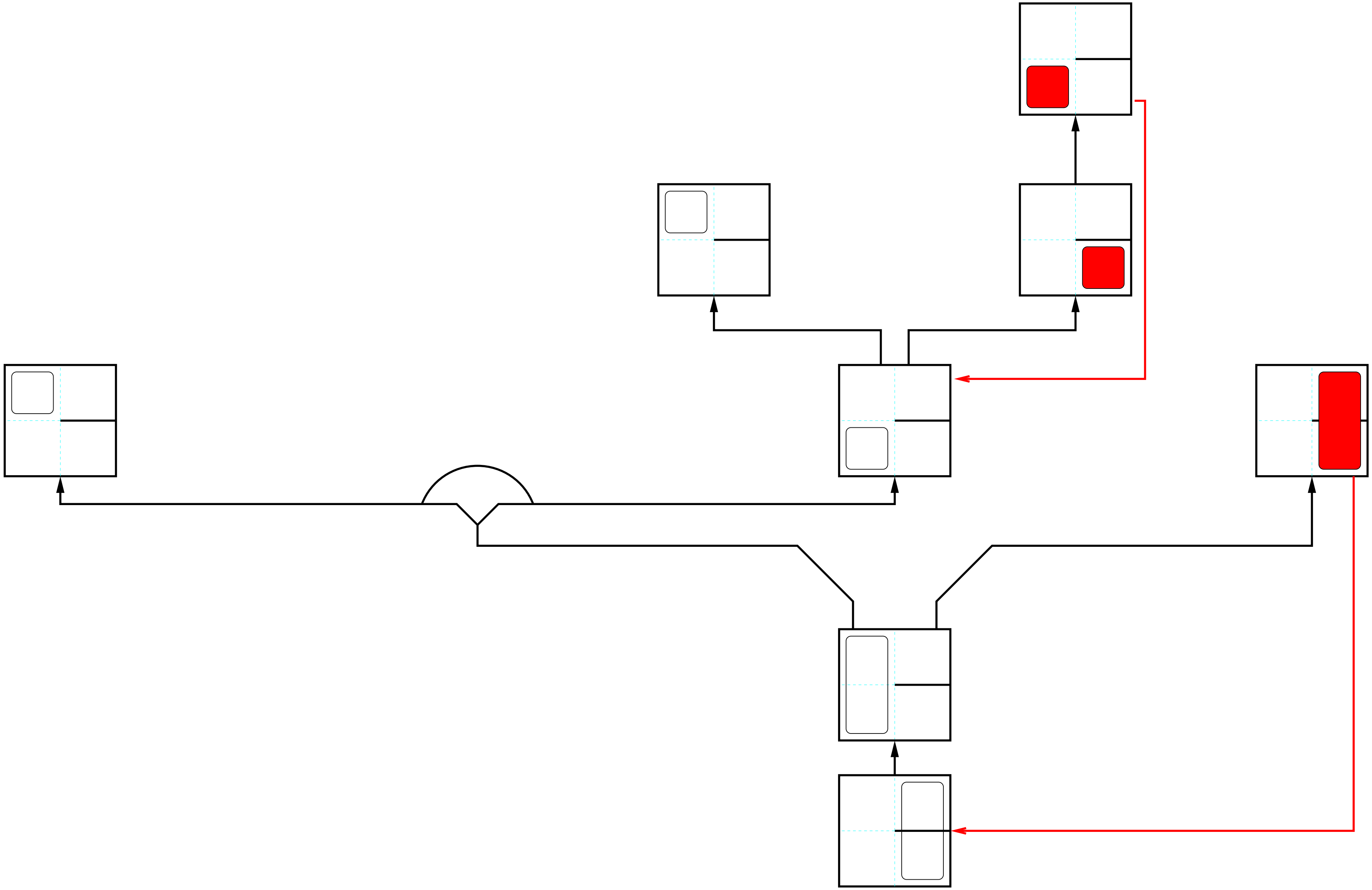


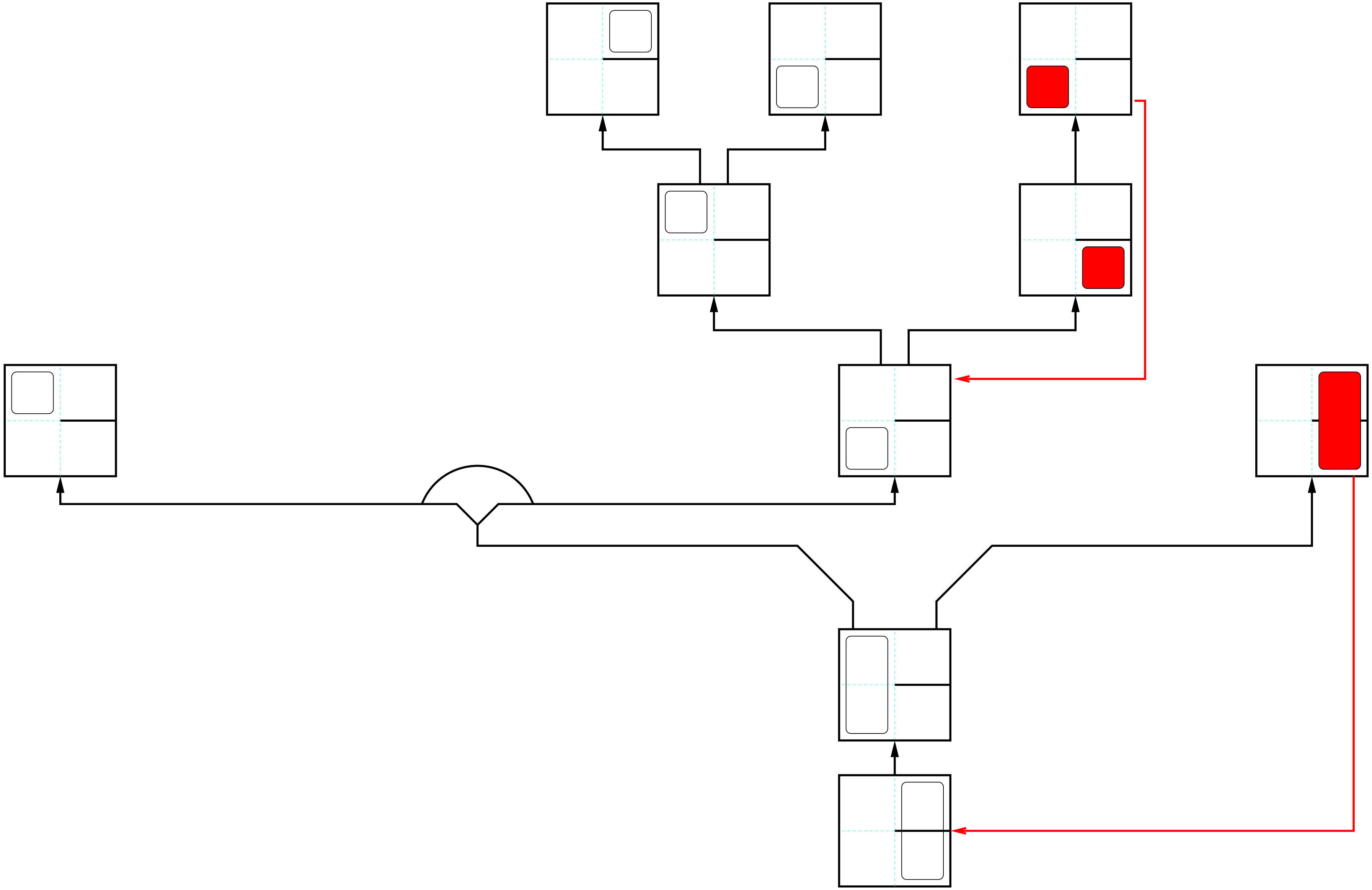


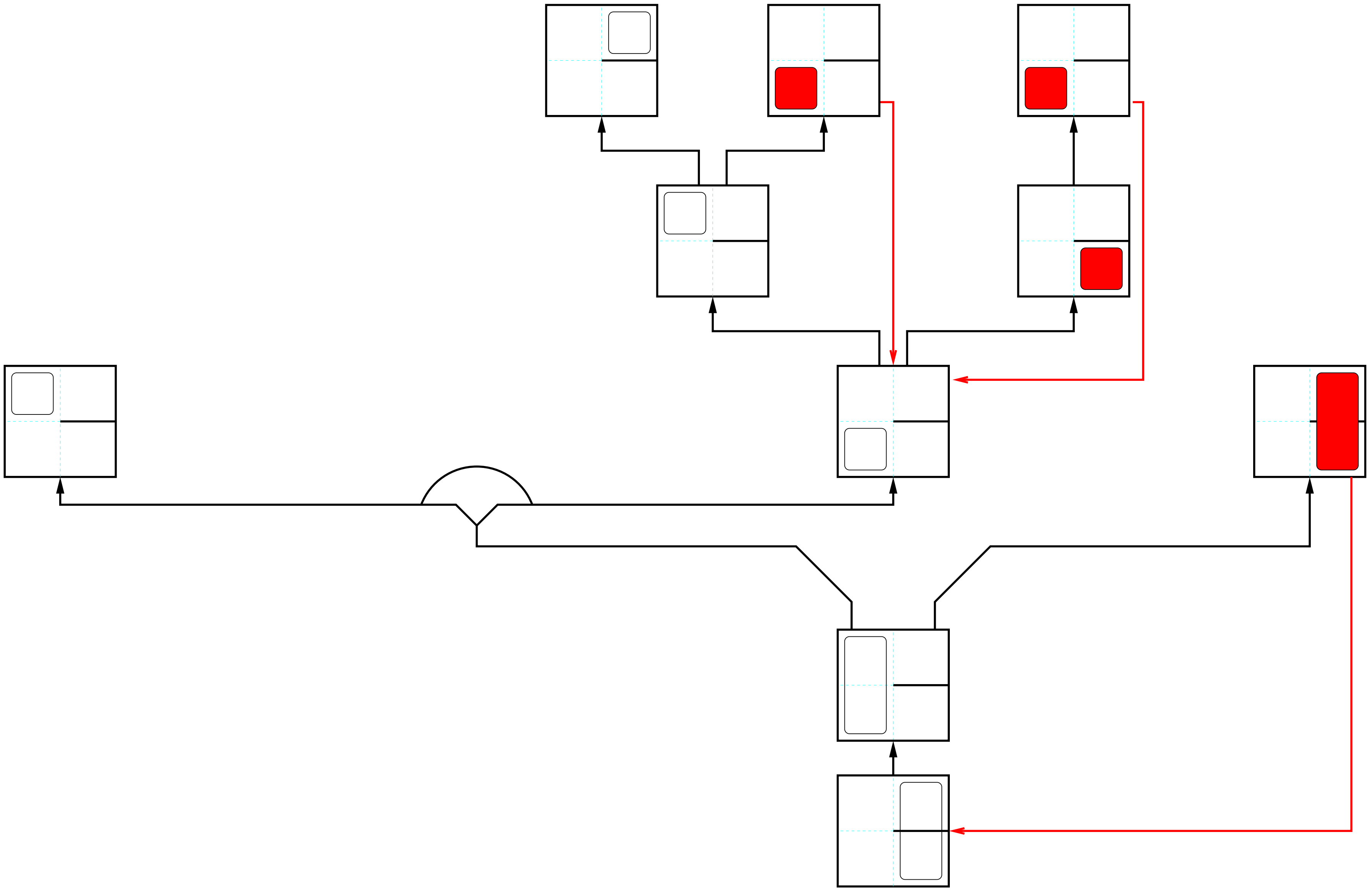


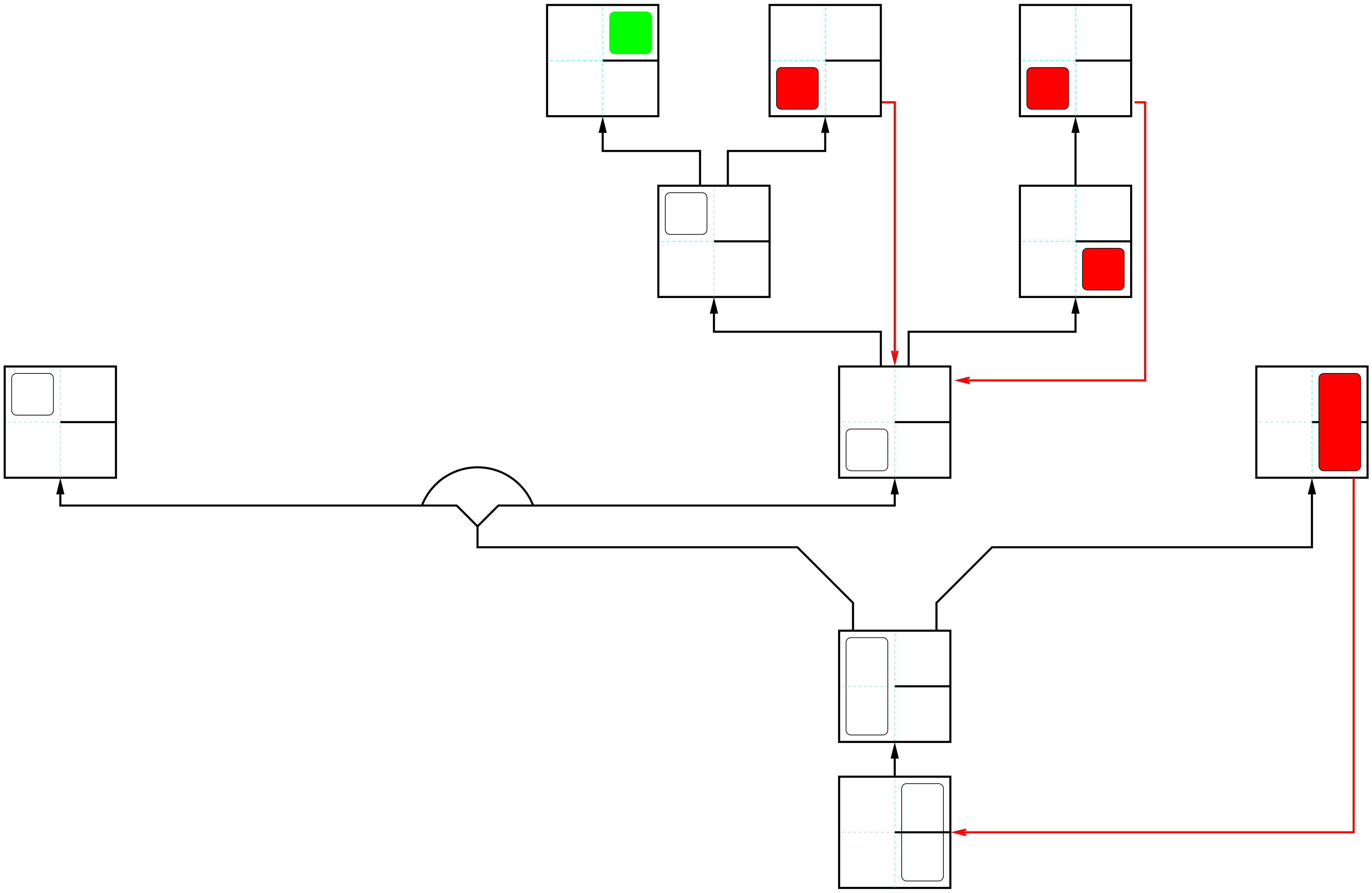


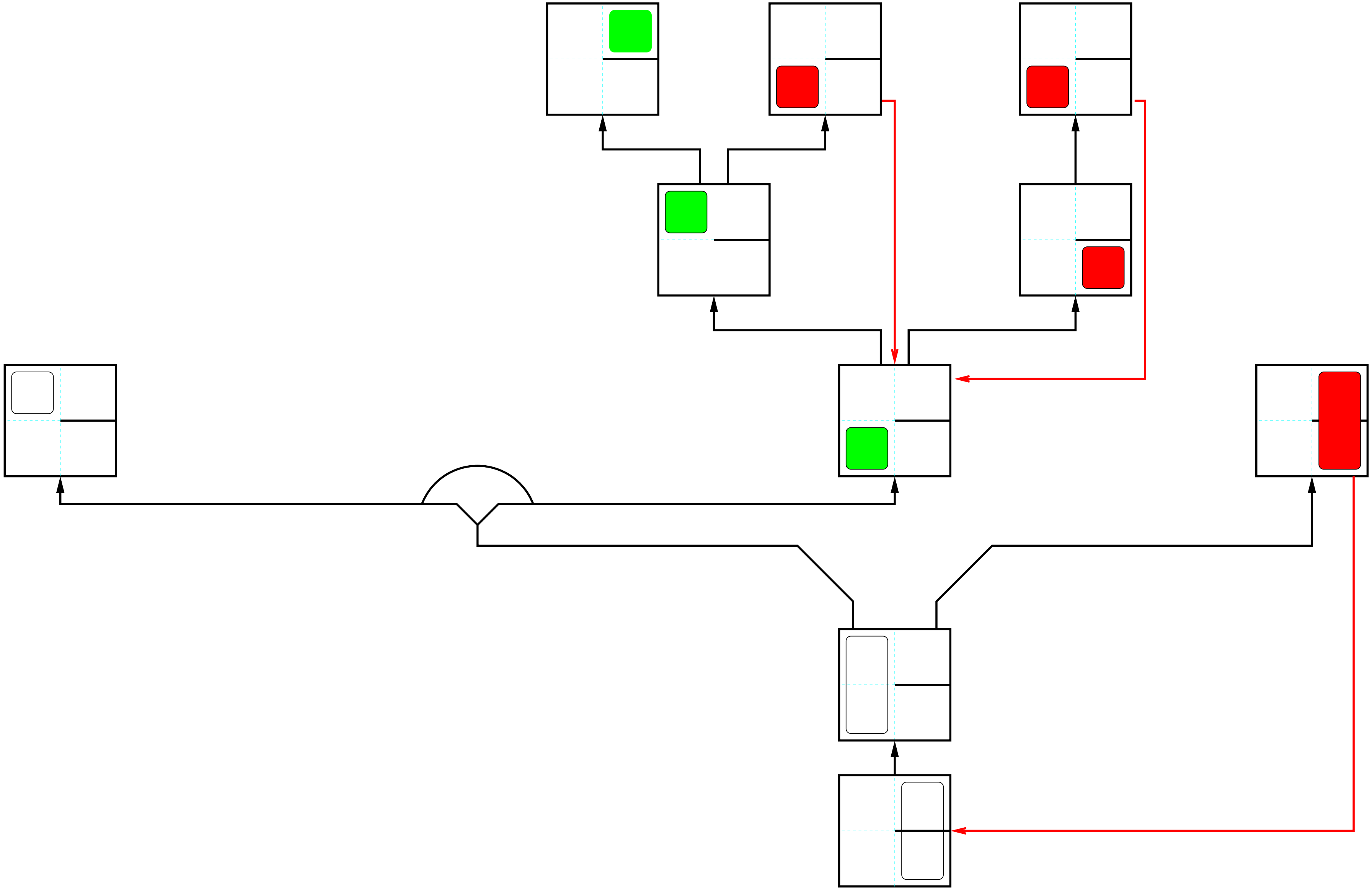


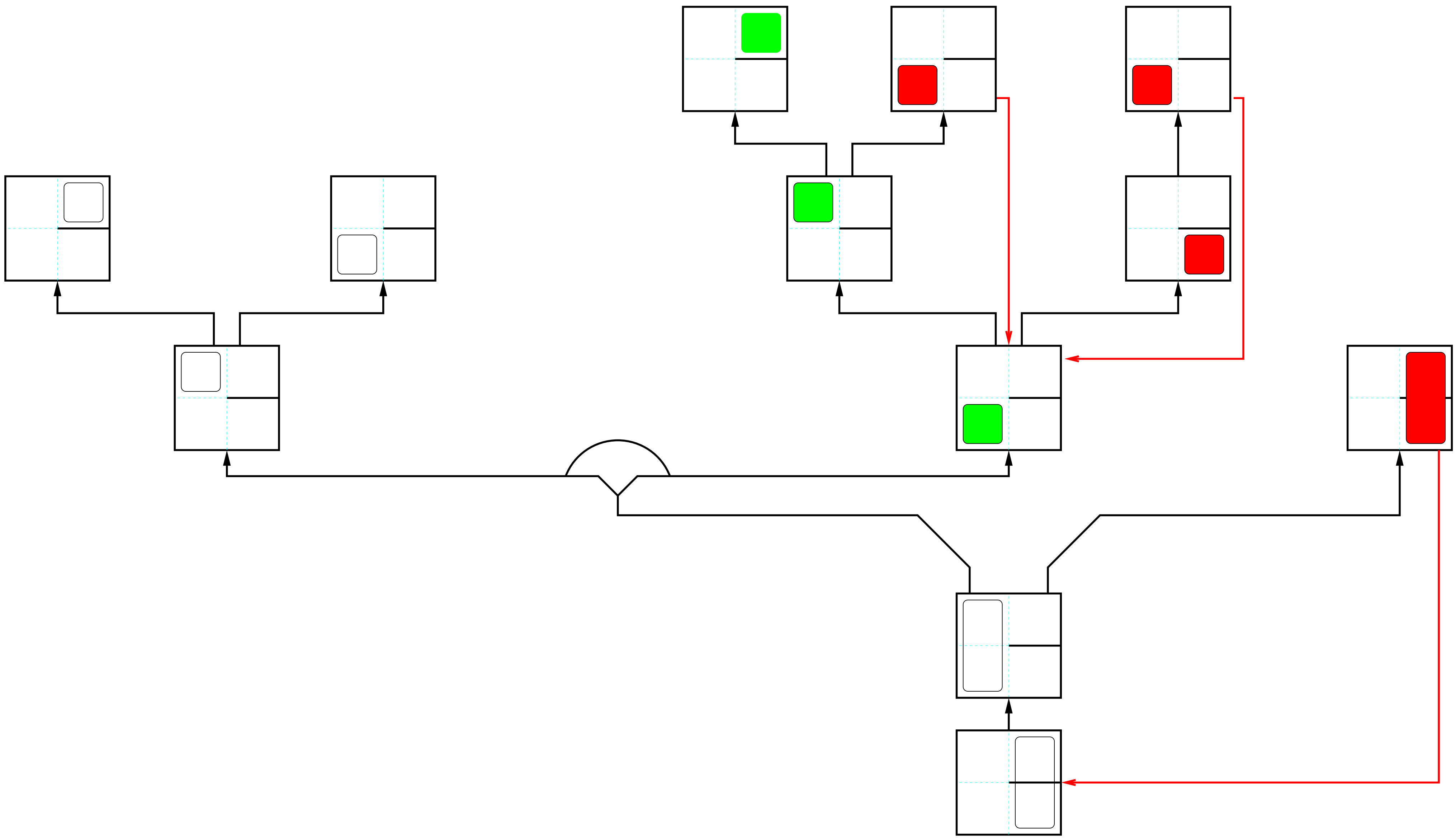


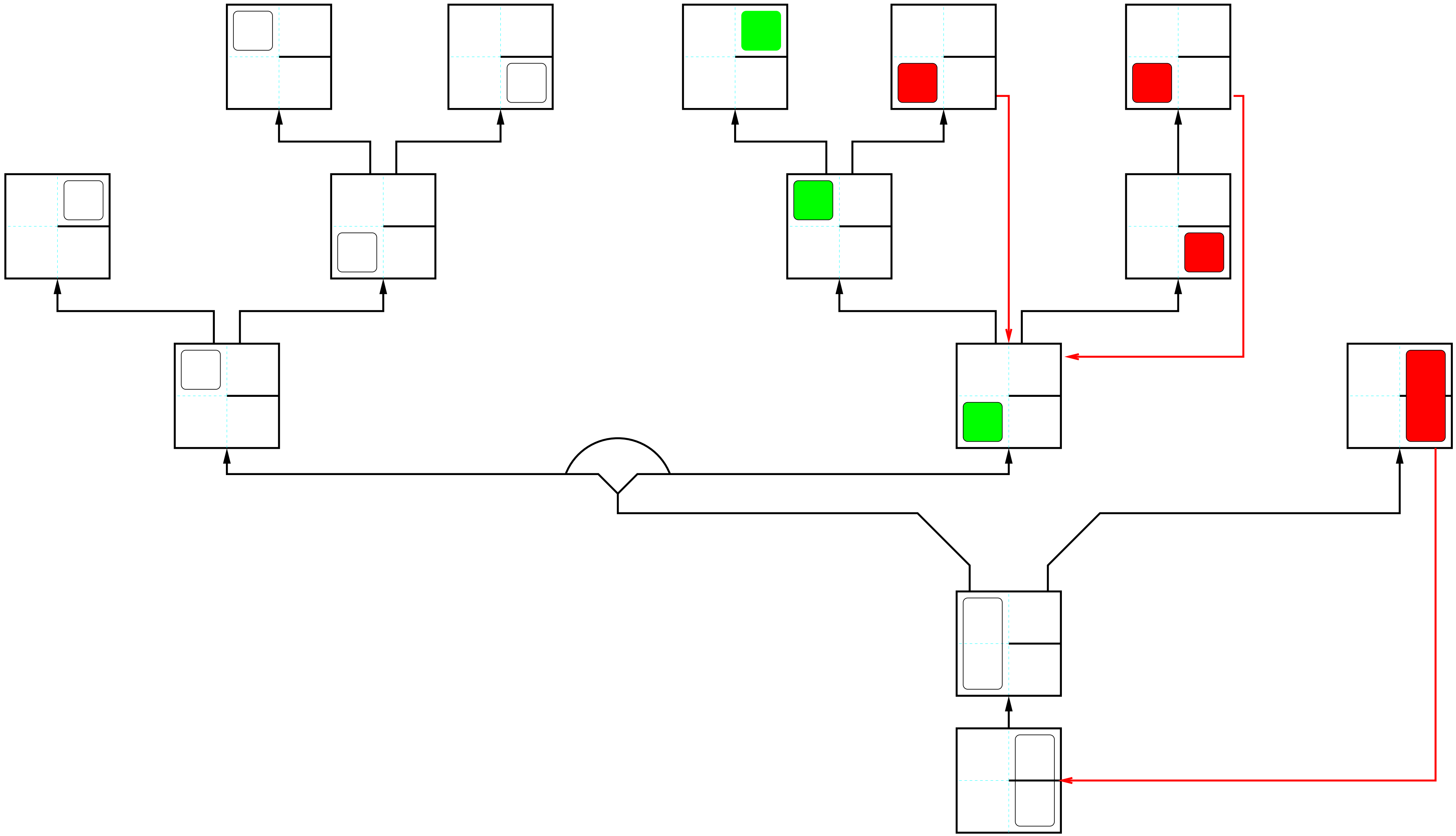


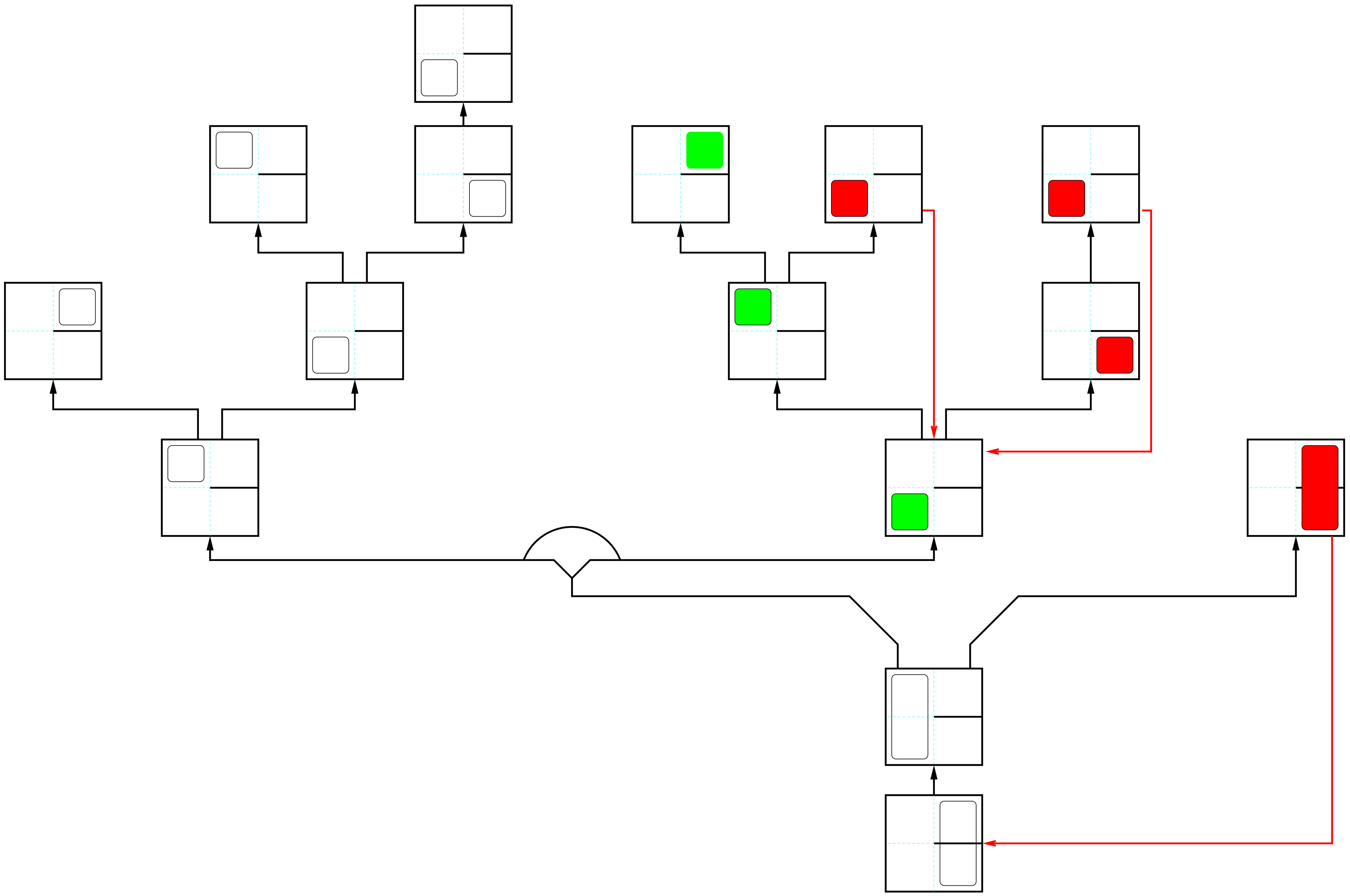


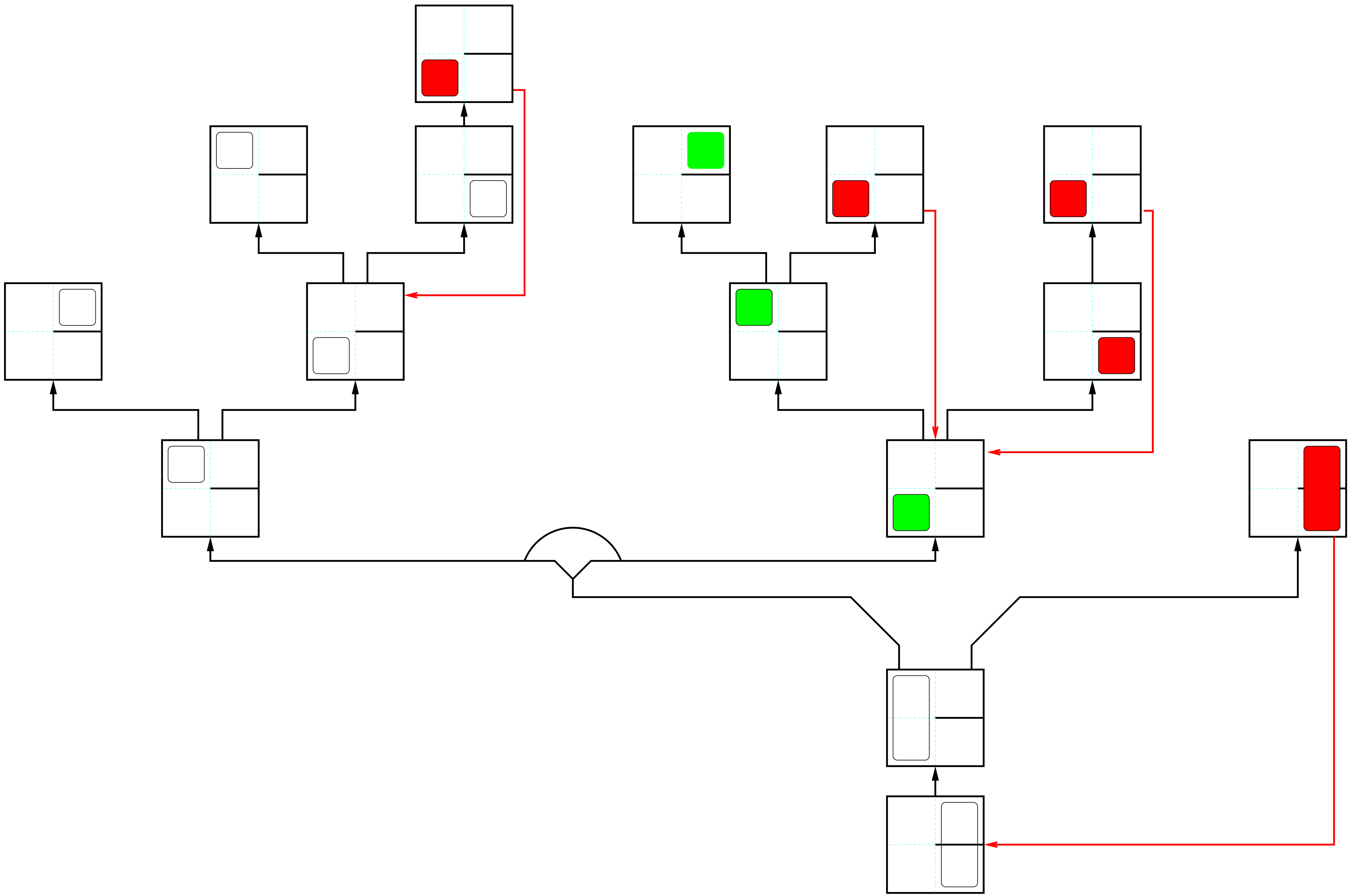


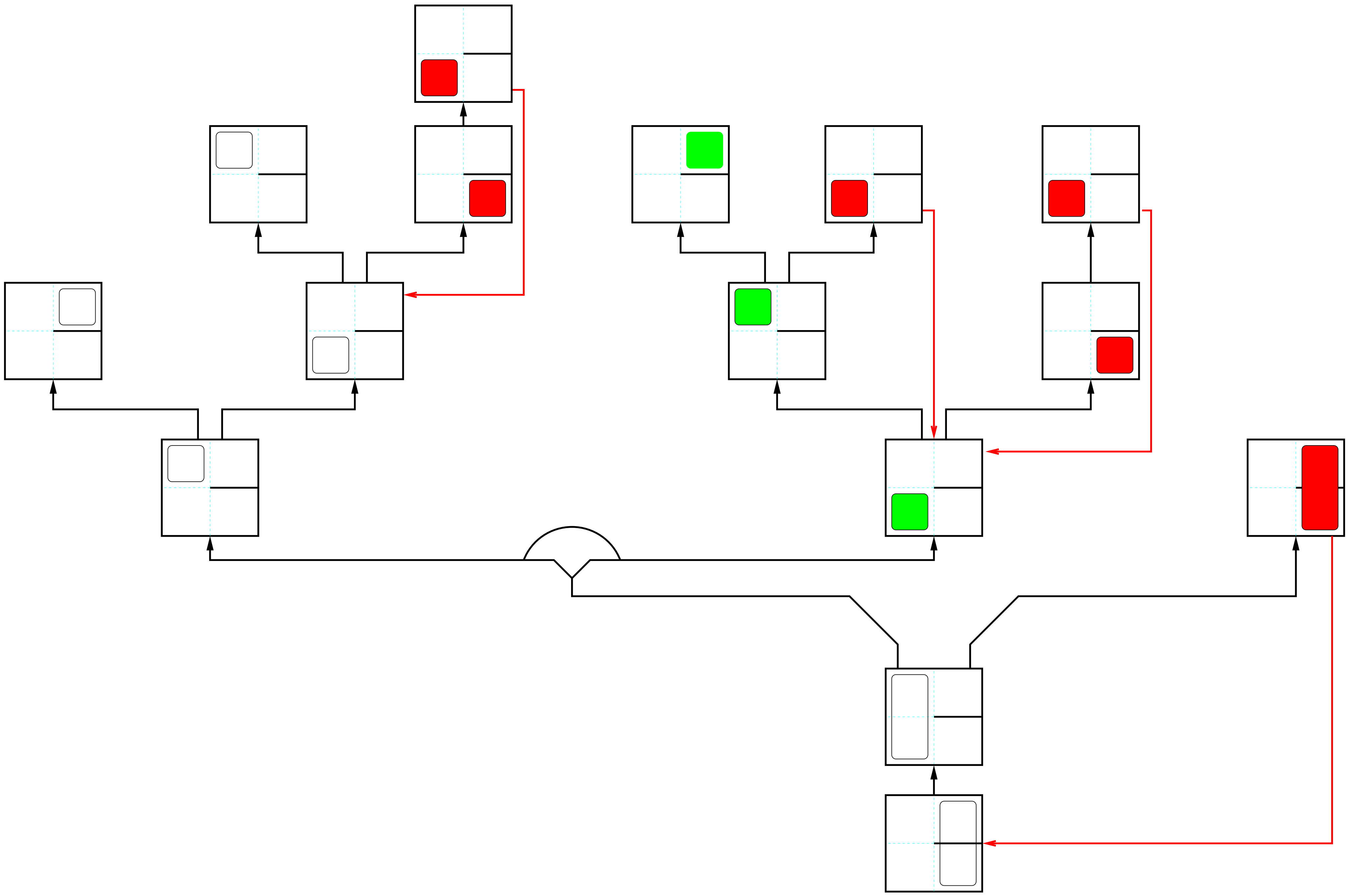


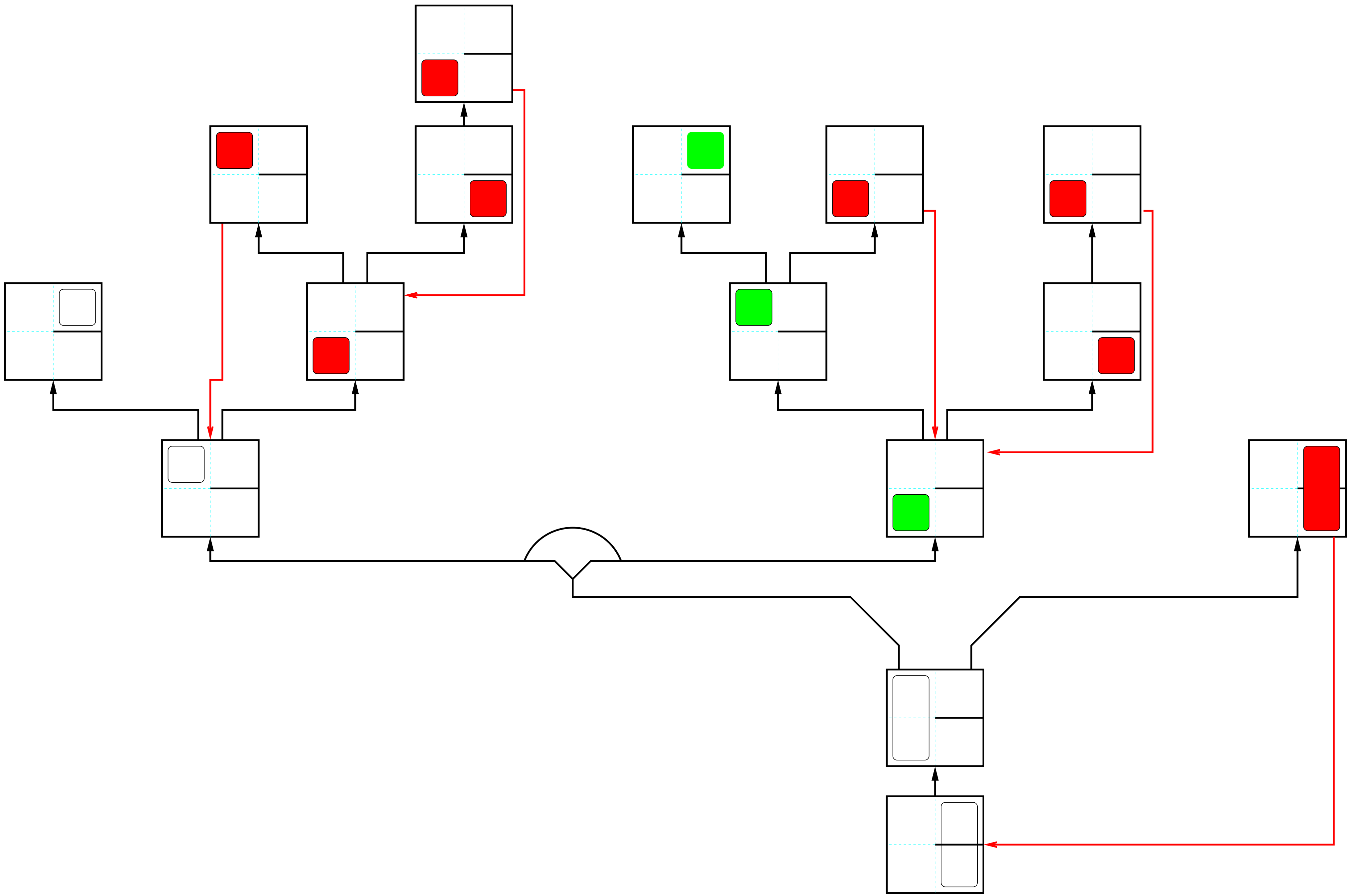


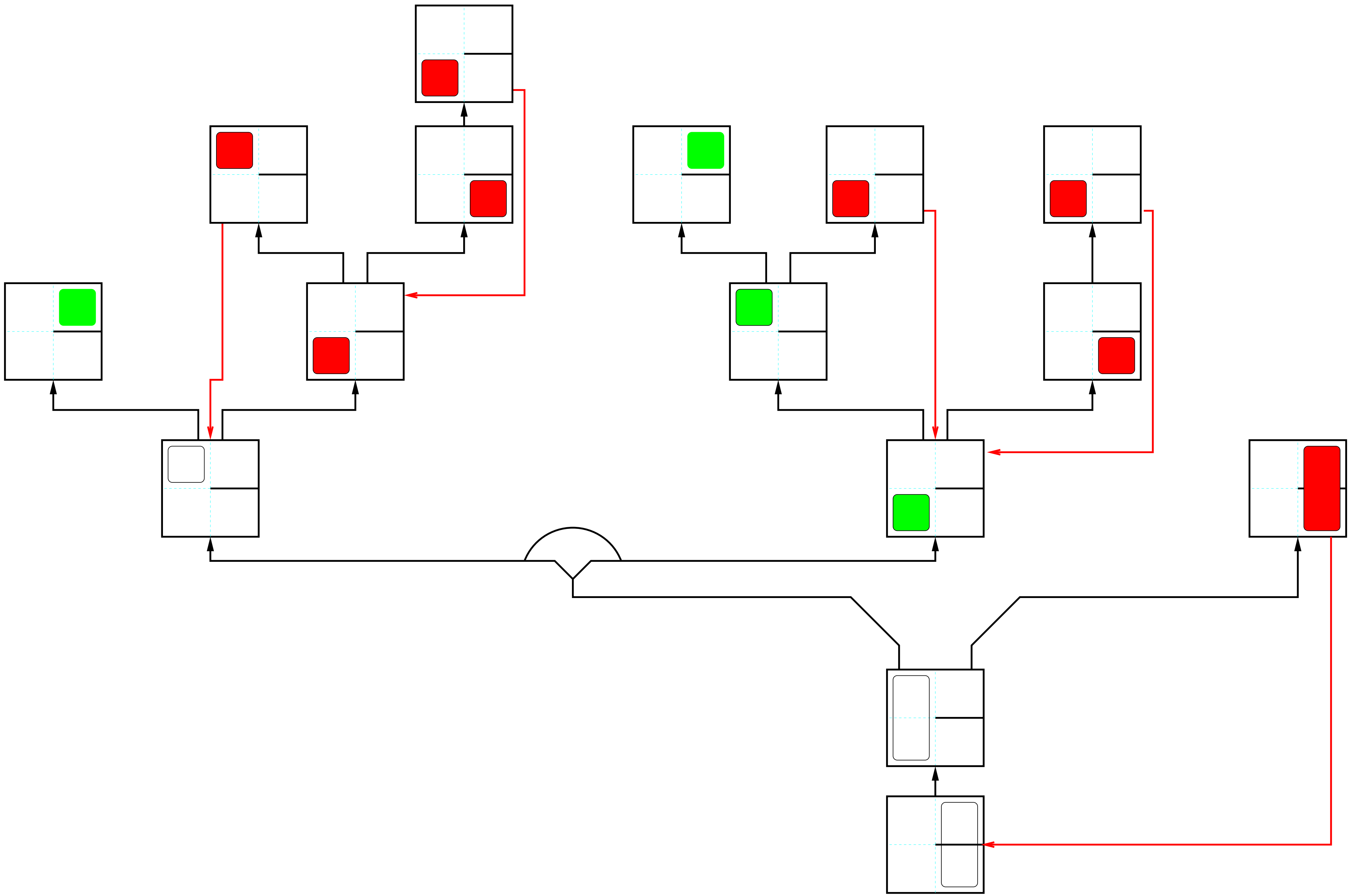


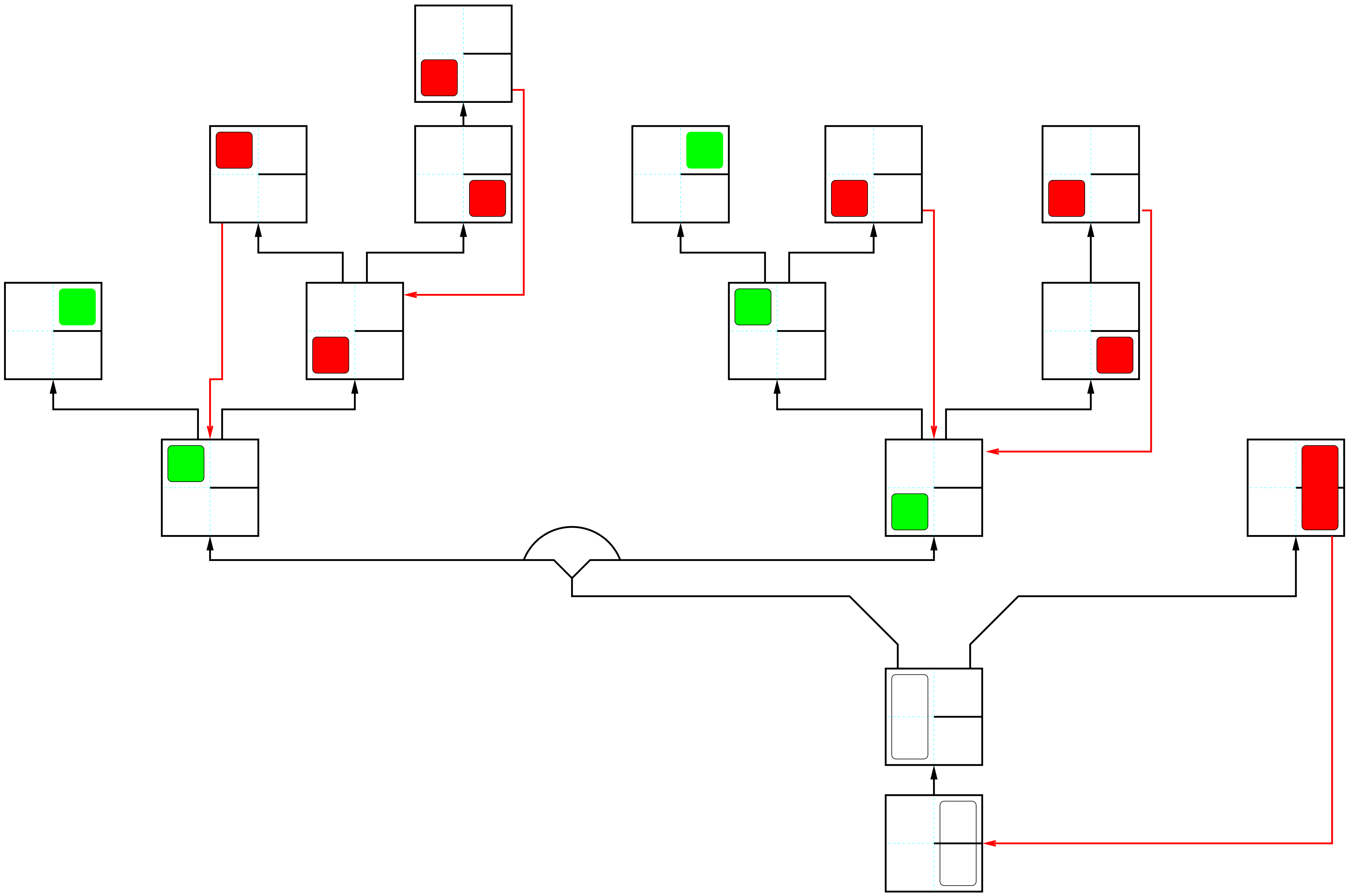


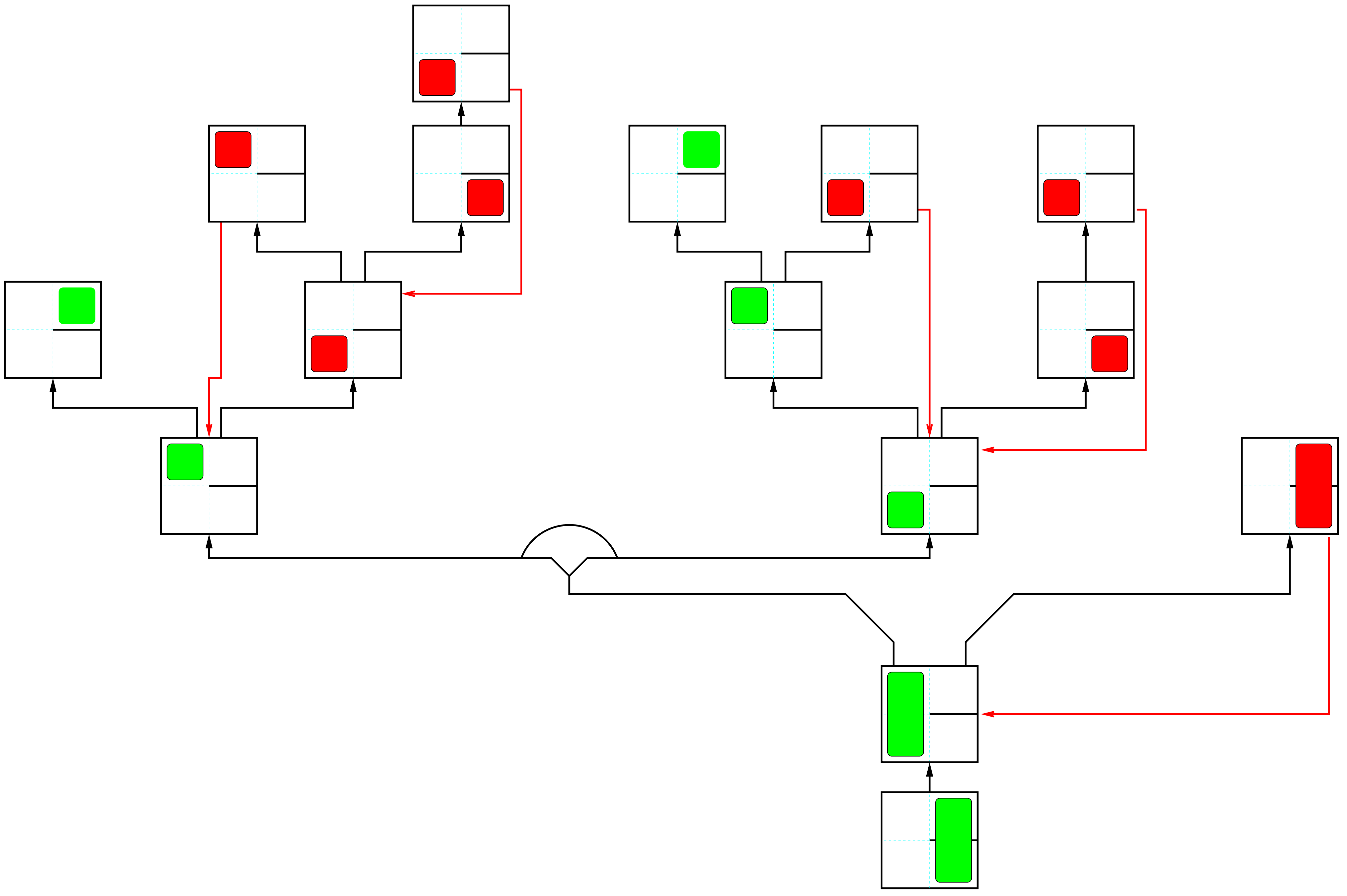












Properties of the Algorithm

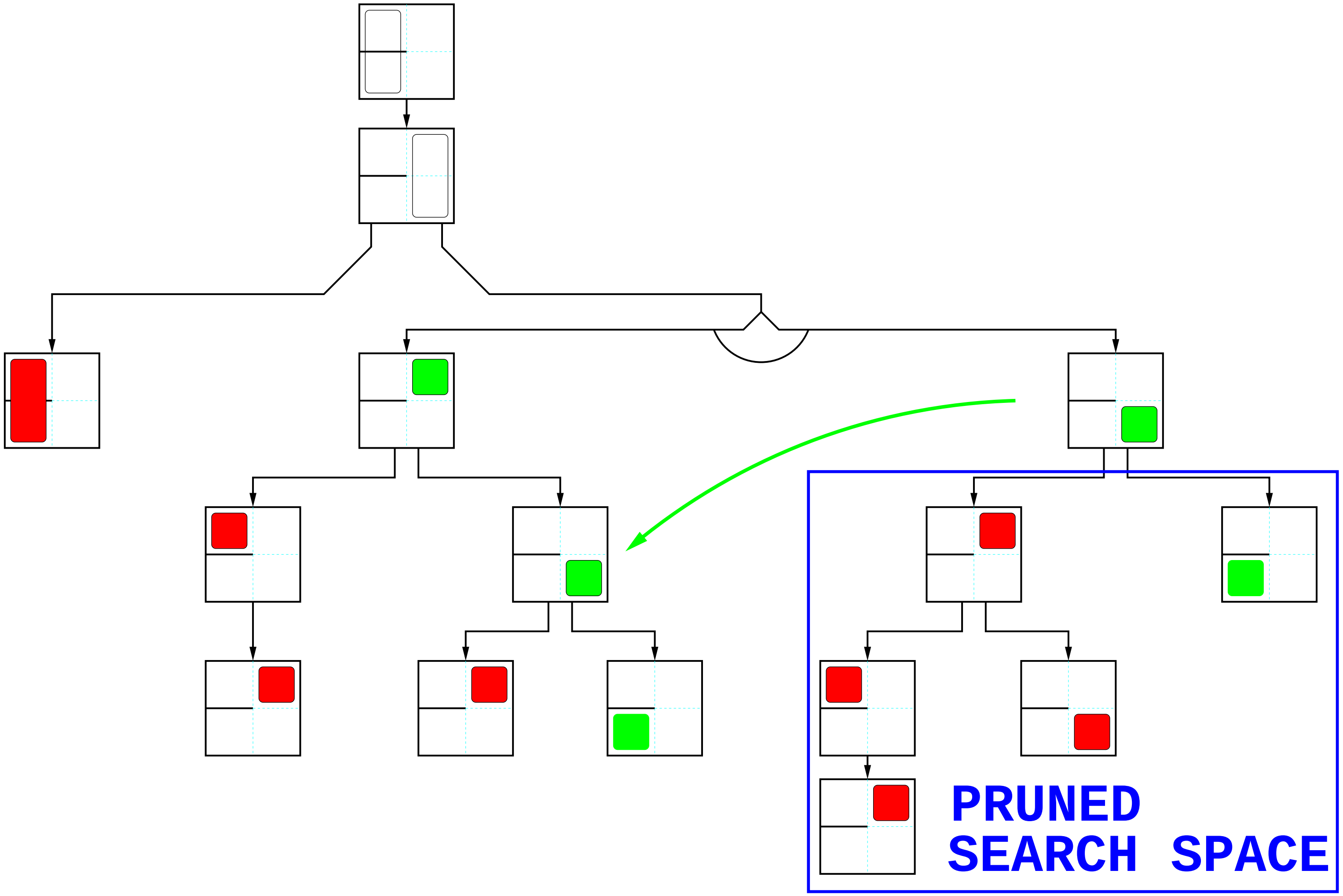
The approach enjoys the following properties:

- Termination: follows from finiteness of acyclic paths.
- If the problem admits a solution, the algorithm returns a solution plan.
- If the problem admits no solution, the algorithm returns with failure.

Other features:

- Plan extracted from traversed state space.
- Preprocessing with Backward Conformant step
- Subplan backward simulation for maximal reuse
- Learning of success
- Learning of failures

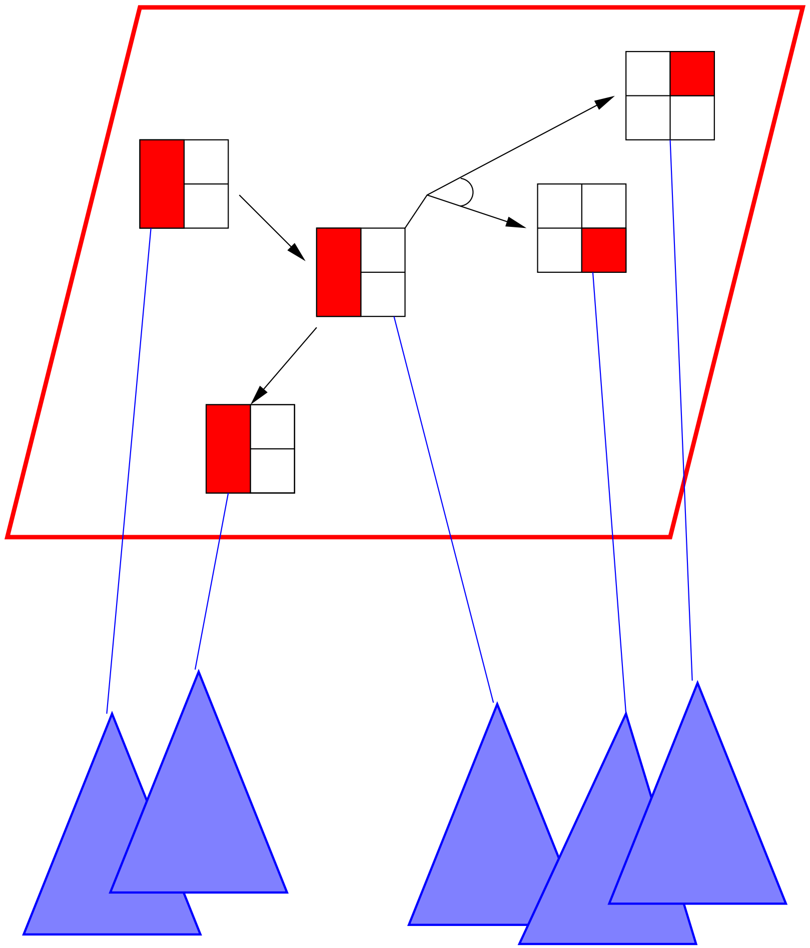
Learning Success



Learning failures

- Learning failures is very useful... but tricky!
- Failures can not be associated to a Belief State, they depend on the **path**.
- Upon failure, associate BS with a “Failure explanation”:
  - a set of BS that will cause again failure if on path;
- A failure explanation identifies an equivalence class over failure paths.
- Upon node re-expansion, check the applicability of stored failure explanations;

AND-OR Search in the Space of BDDs



Use of Symbolic Model Checking Techniques

From Symbolic Model Checking, BDD-based representation of nondeterministic domains.

From Heuristic-Symbolic Conformant Planning:

- BDD-based representation of Belief States, storage with hashing;
- Symbolic Expansion of Belief States under all possible actions;

In addition, we developed

- the data structures for AND-OR grap traversal
- representation of the Observation Relation as BDDs in state, action and observation variables
- expansion of a Belief States under the possible observations
  - relational operations to compute sets of truth and falsity for observation variables

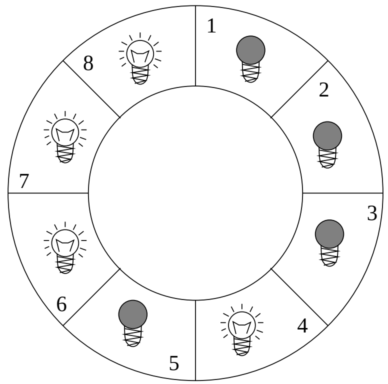
Implementation

MBP, a general planner for nondeterministic domains

- download at `http://sra.itc.it/tools/mbp/`
- built on top of the NuSMV symbolic model checker
- download at `http://nusmv.irst.itc.it/`

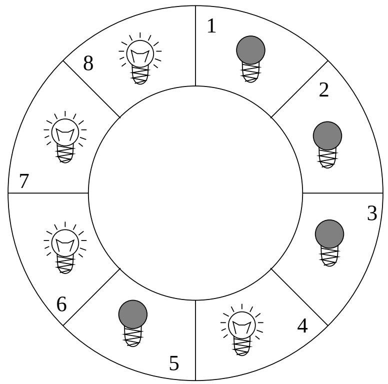
Experimental evaluations in [JAIR'00], [IJCAI'01a,b], [ICAPS'02], [AIJ'04].

More on knowledge goals...



- Positioning goal:
  - Uncertainty on initial position, known position of lights
  - find plan π such that, after execution, the status of world is completely known, i.e.
  - $\mathcal{K}(\texttt{Loc} = l)$ holds for some l
  - what if environment is dynamic (e.g. can turn off lights)?

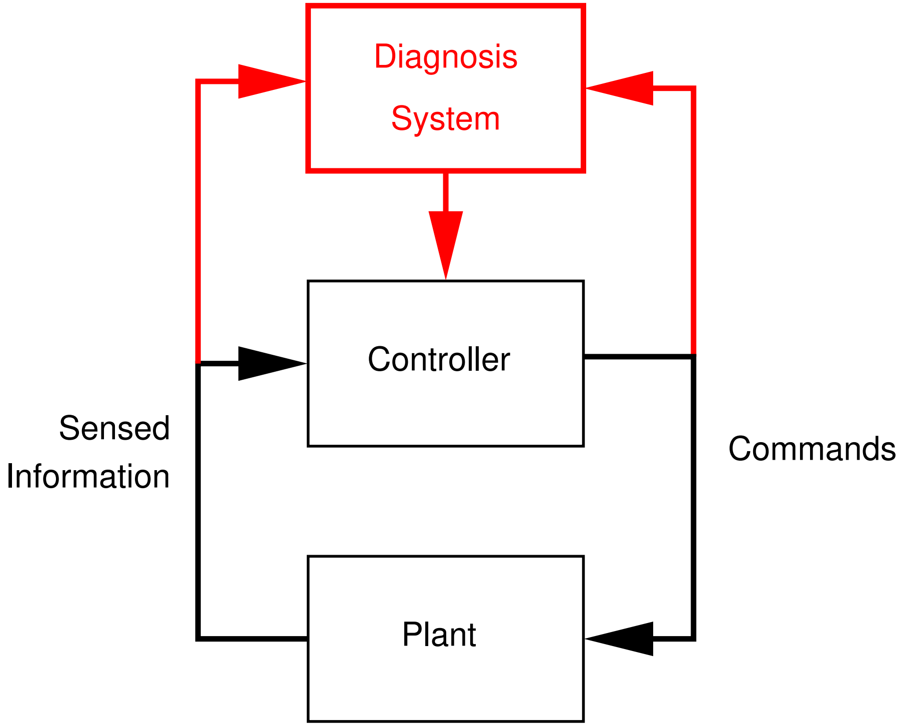
More on knowledge goals...



- object-level goal G and knowledge-level K G may not coincide!
  - `Light[5]` $= on$ vs $\mathcal{K}($`Light[5]` $= on)$
  - Plan: solve positioning problem, then go to 5, and turn on.
  - with dynamic environment (e.g. can turn off lights)
  - positioning problem may become unsolvable!!!
  - Plan: visit all roooms, and turn on the light if off.
  - `Light[5]` $= on$ solvable
  - but $\mathcal{K}($`Light[5]` $= on)$

Other problems (to be discussed offline)

The Diagnosability Problem

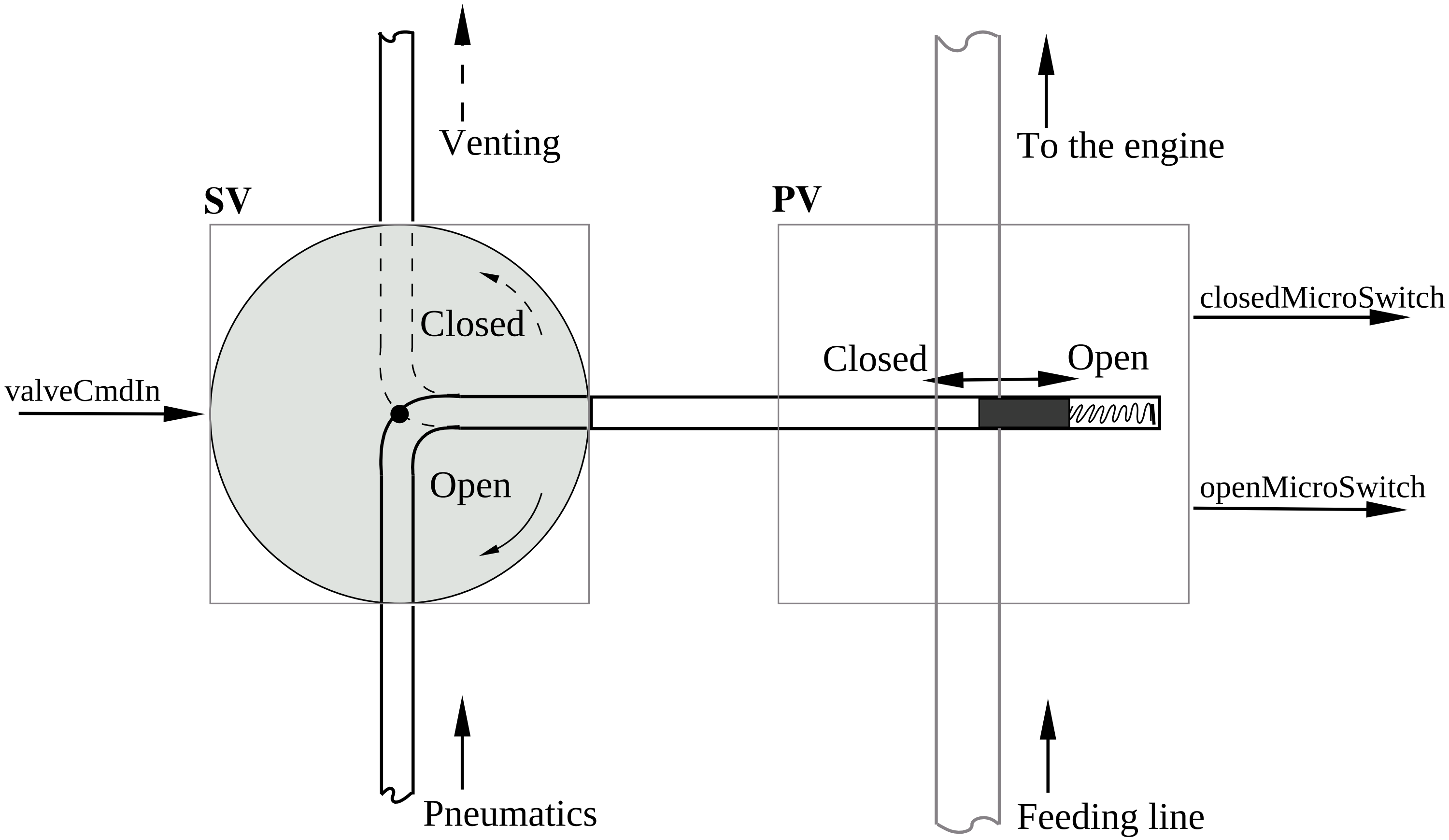


The diagnosis system tracks the hidden state of the plant *over time*.

The diagnosis process has no direct control over the control loop.

Diagnosability: verify *off-line* that an ideal Diagnosis System can infer *at run-time* accurate and sufficient information on the behavior of the observed Plant.

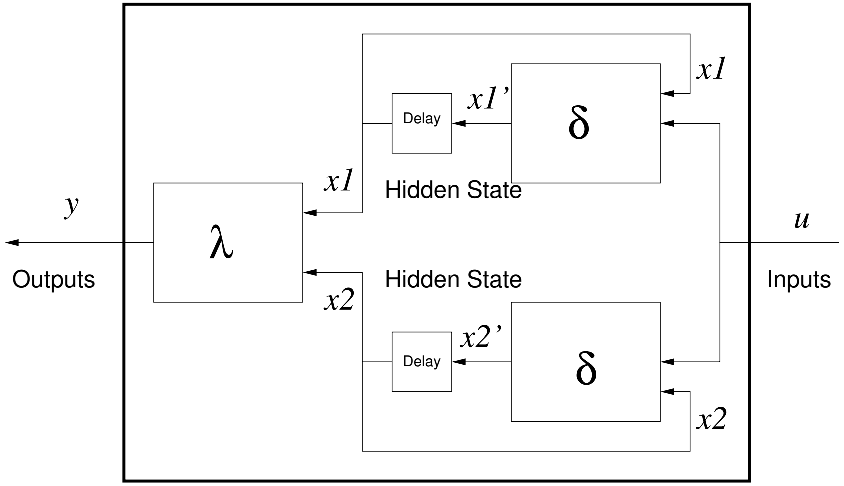
A Simple Example: Flow Control Valve

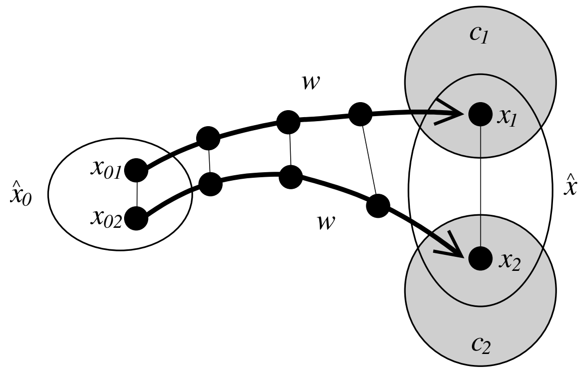


Diagnosability as Coupled Reachability

Coupled Twin Plant of P: $P \cdot P = \langle X \cdot X, U, Y, \delta \cdot \delta, \lambda \cdot \lambda \rangle$

- $X \cdot X \subset X^2$, and $(x_1, x_2) \in X \cdot X$ iff there exists $y \in Y$ such that x_1/y and x_2/y;
- $((x_1, x_2), u, (x'_1, x'_2)) \in \delta \cdot \delta$ iff $x_1 \xrightarrow{u} x'_1$ and $x_2 \xrightarrow{u} x'_2$;
- $((x_1, x_2), y) \in \lambda \cdot \lambda$ iff x_1/y and x_2/y.





$\sigma_1 : x_{01} \xrightarrow{w} x_1$ and $\sigma_2 : x_{02} \xrightarrow{w} x_2$ are two feasible executions in P iff $\sigma_{12} : (x_{01}, x_{02}) \xrightarrow{w} (x_1, x_2)$ is a feasible execution in $P \cdot P$.

Model Checking of Multi-Agent Systems

- Computation Tree Logic extended with K
- agent systems as hierarchical Kripke structures
- temporal and knowledge accessibility relations represented as transition relations
- a preliminary implementation: the BDD-based model checker NuMAS
- jww Massimo Benerecetti

Conclusions

- Reasoning about dynamic systems
  - incompletely specified initial condition
  - nondeterministic actions/dynamics
  - an important subcase
- The need for analysis at knowledge level.
  - problem formulation in belief space
- Effective use of boolean techniques
  - Binary Decision Diagrams
  - Propositional SAT very promising...