Abstraction and Refinement

State-space explosion

In practice, model-checking encounters the problem of state-space explosion:

due to data: var with n bits $\rightarrow 2^n$ states

due to concurrency: n parallel components with $n!$ different orderings

Countermeasures:

Compression: efficient representations (e.g. BDDs)

Reduction: find a simpler, equivalent problem

Abstraction: identify and ignore “unimportant” information

Example 1 (loop)

Consider the following program with three numeric variables x, y, z.

ℓ_1: `y = x+1;`

ℓ_2: `z = 0;`

ℓ_3: `while (z < 100) z = z+1;`

ℓ_4: `if (y < x) error;`

Question: Is the error location reachable?

Example 2 (Sorting)

Another program with three numeric variables x, y, z.

ℓ_1: `if x > y then  swap x,y  else  skip;`

ℓ_2: `if y > z then  swap y,z  else  skip;`

ℓ_3: `if x > y then  swap x,y  else  skip;`

ℓ_4: `skip`

Assumption: initially, x, y, z are all different

Question: Are x, y, z sorted in ascending order when reaching ℓ_4?

Example 3 (Device driver)

C code for Windows device driver

Operations on a semaphor: lock, release

Lock and release must be used alternatingly

Abstraction

Idea: throw away (abstract from) “unimportant” information

Handling *infinite* state spaces

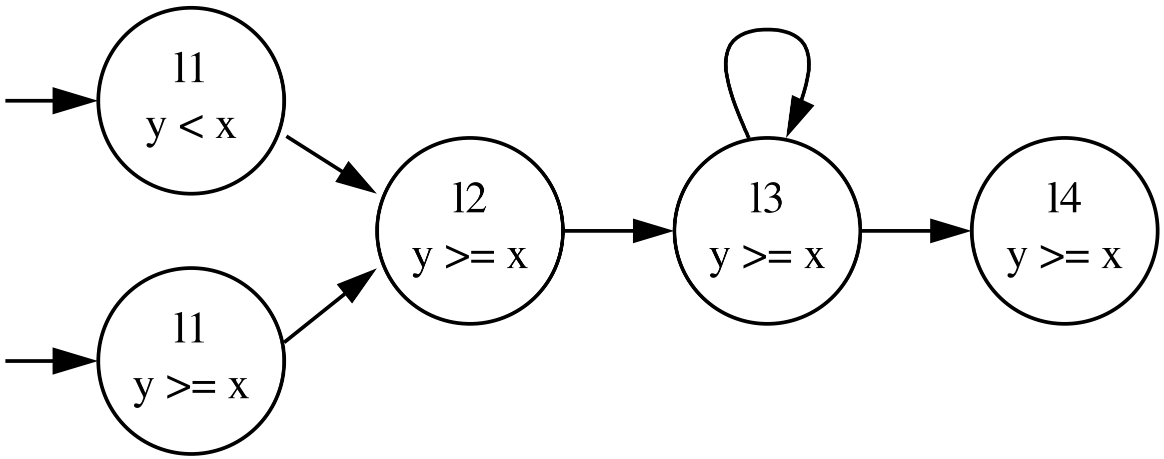
Reduce (large) finite problems to smaller ones

Alternative point of view: merge “equivalent” states

Example 1

Omit concrete values of `x,y,z`; retain only the following information: program counter, predicate $y < x$

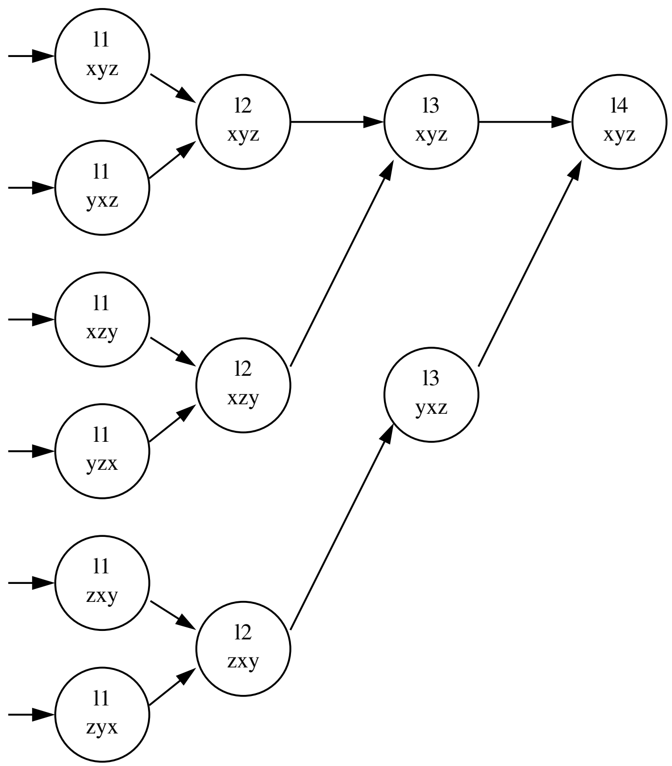
Resulting (abstract) Kripke structure:



Result: ℓ_4 is reachable only with $y \geq x$; the error will not happen.

Example 2

Omit concrete values of `x`, `y`, `z`; retain only program counter and permutation of *x*, *y*, *z*



Result: ℓ_4 is reachable only with *xyz*; no error.

Questions: What is the logical relation between the original programs and their abstract versions? What do the abstract versions really say about the original programs?

In Example 1, the error is unreachable in both the original and the abstract version.

However, in Example 1, the original structure terminates but the abstract version does not.

Which conditions must hold for the abstract structure in order to draw meaningful conclusions about the original structure?

Simulation

Let $\mathcal{K}_1 = (S, \to_1, s_0, AP, \nu)$ and $\mathcal{K}_2 = (T, \to_2, t_0, AP, \mu)$ be two Kripke structures (S, T are possibly *infinite*), and let $H \subseteq S \times T$ be a relation.

H is called a simulation from $\mathcal{K}_1$ to $\mathcal{K}_2$ iff

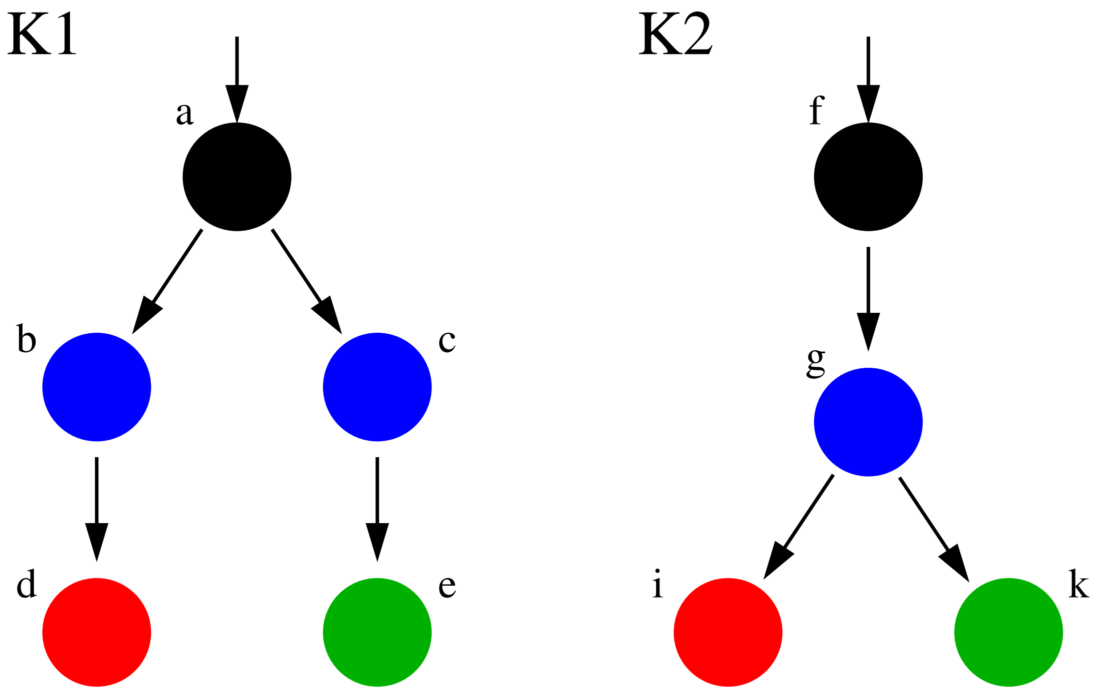
(i) $(s_0, t_0) \in H$;

(ii) for all $(s, t) \in H$ we have: $\nu(s) = \mu(t)$;

(iii) if $(s, t) \in H$ and $s \to_1 s'$, then there exists t' such that $t \to_2 t'$ and $(s', t') \in H$.

We say: $\mathcal{K}_2$ simulates $\mathcal{K}_1$ (written $\mathcal{K}_1 \leq \mathcal{K}_2$) if such a simulation H exists.

Intuition: $\mathcal{K}_2$ can do anything that is possible in $\mathcal{K}_1$.



$\mathcal{K}_2$ simulates $\mathcal{K}_1$ (with $H = \{(a, f), (b, g), (c, g), (d, i), (e, k)\}$).

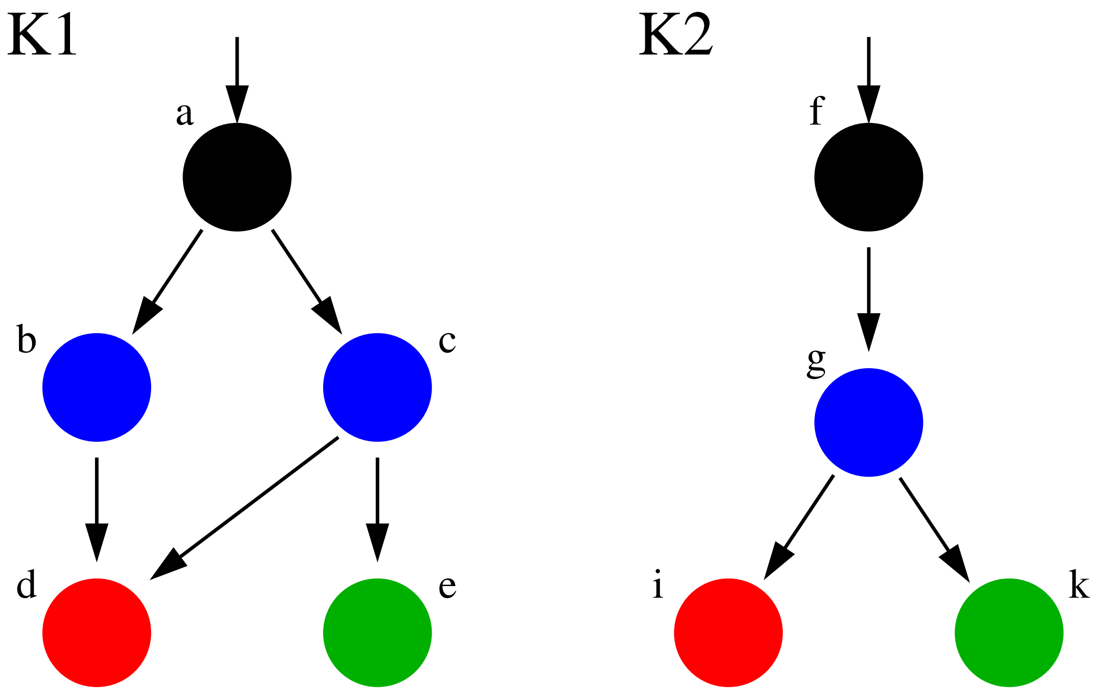
However,: $\mathcal{K}_1$ does *not* simulate $\mathcal{K}_2$!

Bisimulation

A relation H is called a bisimulation between $\mathcal{K}_1$ and $\mathcal{K}_2$ iff H is a simulation from $\mathcal{K}_1$ to $\mathcal{K}_2$ and $\{ (t, s) \mid (s, t) \in H \}$ is a simulation from $\mathcal{K}_2$ to $\mathcal{K}_1$.

We say: $\mathcal{K}_1$ and $\mathcal{K}_2$ are bisimilar (written $\mathcal{K}_1 \equiv \mathcal{K}_2$) iff such a relation H exists.

Careful: In general, $\mathcal{K}_1 \leq \mathcal{K}_2$ and $\mathcal{K}_2 \leq \mathcal{K}_1$ do *not* imply $\mathcal{K}_1 \equiv \mathcal{K}_2$!



(Bi-)Simulation and model checking

Let $\mathcal{K}_1 \leq \mathcal{K}_2$ and ϕ an LTL formula. Then:

$\mathcal{K}_2 \models \phi$ implies $\mathcal{K}_1 \models \phi$ (for universal model checking).

The other direction is not guaranteed!

Existential abstraction

Let $\mathcal{K} = (S, \rightarrow, r, AP, \nu)$ be a Kripke structure (concrete structure).

Let $\approx$ be an equivalence relation on S such that for all $s \approx t$ we have $\nu(s) = \nu(t)$ (we say: $\approx$ respects ν).

Let $[s] := \{\, t \mid s \approx t \,\}$ denote the equivalence class of s;
$[S]$ denotes the set of all equivalence classes.

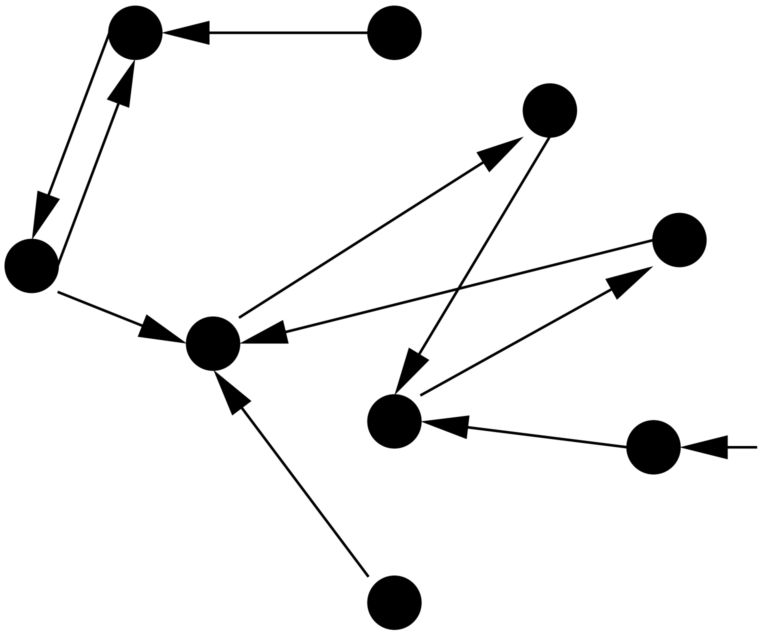
The abstraction of S w.r.t. $\approx$ denotes the structure $\mathcal{K}' = ([S], \rightarrow', [r], AP, \nu')$, where

$[s] \rightarrow' [t]$ for all $s \rightarrow t$;

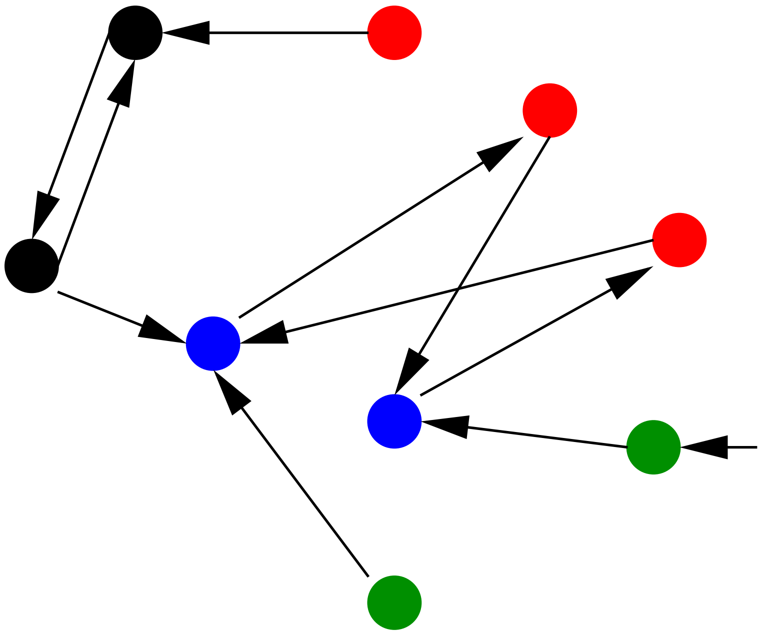
$\nu'([s]) = \nu(s)$ (this is well-defined!).

Example

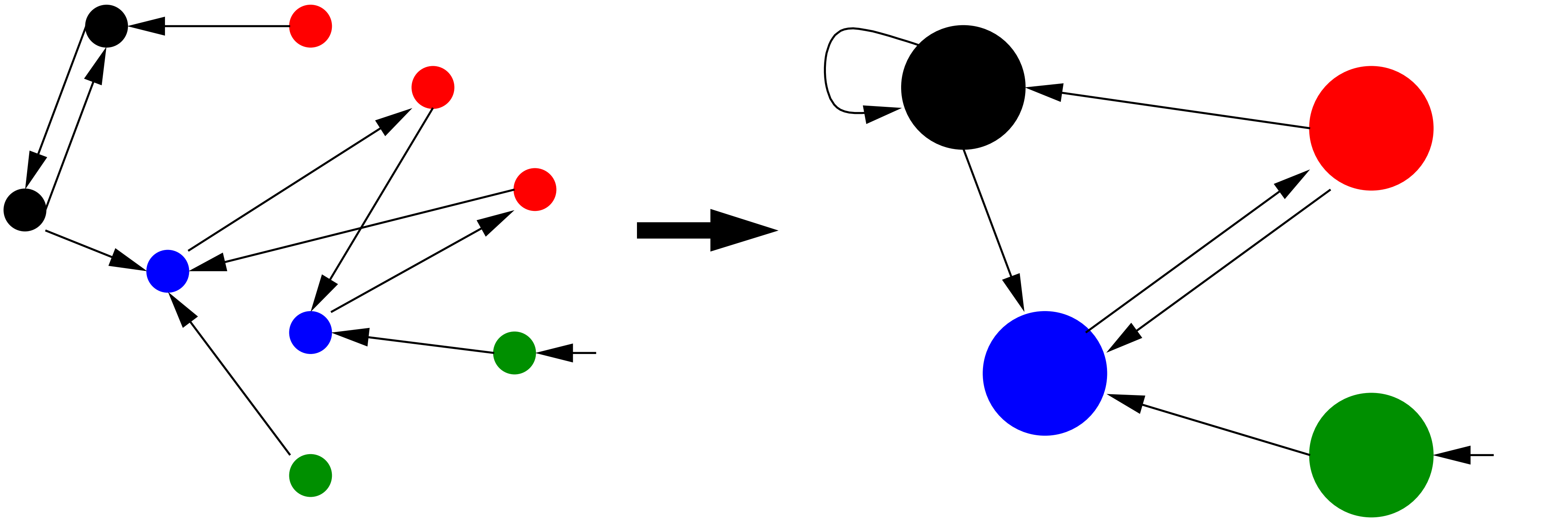
Consider the Kripke structure below:



States partitioned into equivalence classes:



Abstract structure obtained by quotienting:

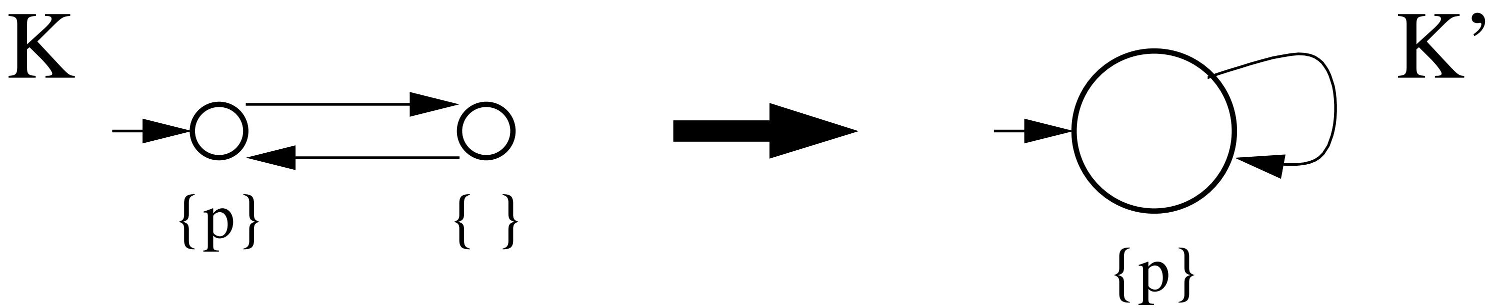


Let $\mathcal{K}'$ be a structure obtained by abstraction from $\mathcal{K}$.

Then $\mathcal{K} \leq \mathcal{K}'$ holds (simulation relation: $\{\, (s, [s]) \mid s \in S \,\}$)

Thus, if $\mathcal{K}'$ satisfies some LTL formula, so does $\mathcal{K}$.

What happens if $\approx$ does *not* respect ν?

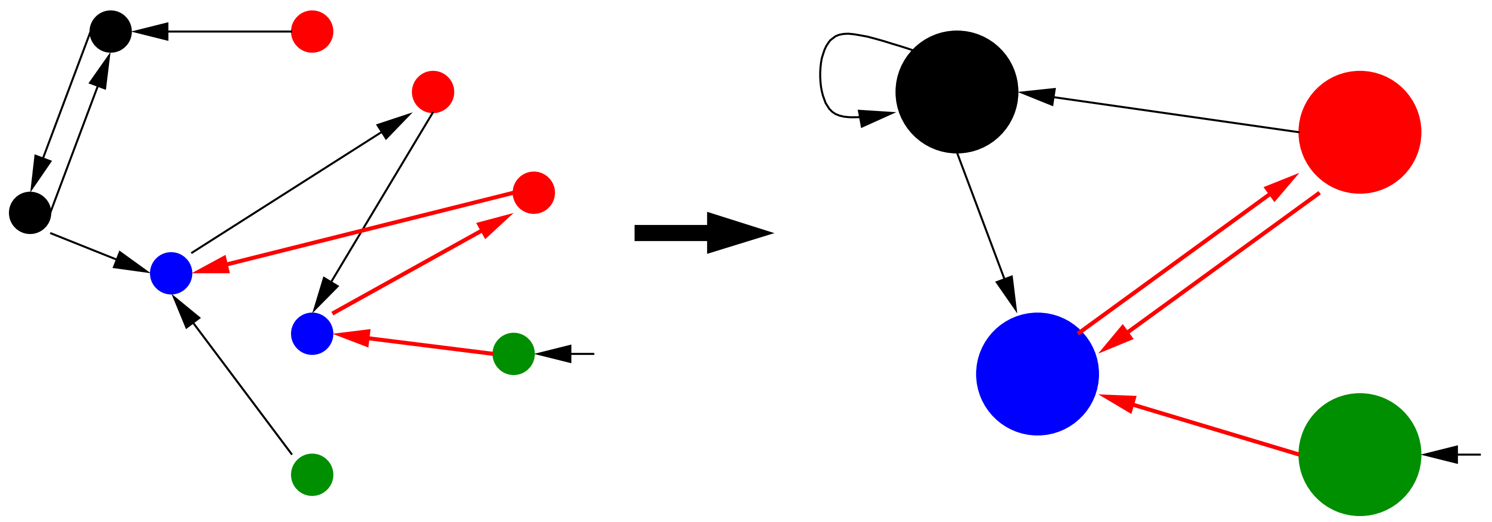


Then $\mathcal{K} \not\preceq \mathcal{K}'$ does not hold.

Example: The abstraction satisfies $\mathbf{G}\ p$, the concrete system does not.

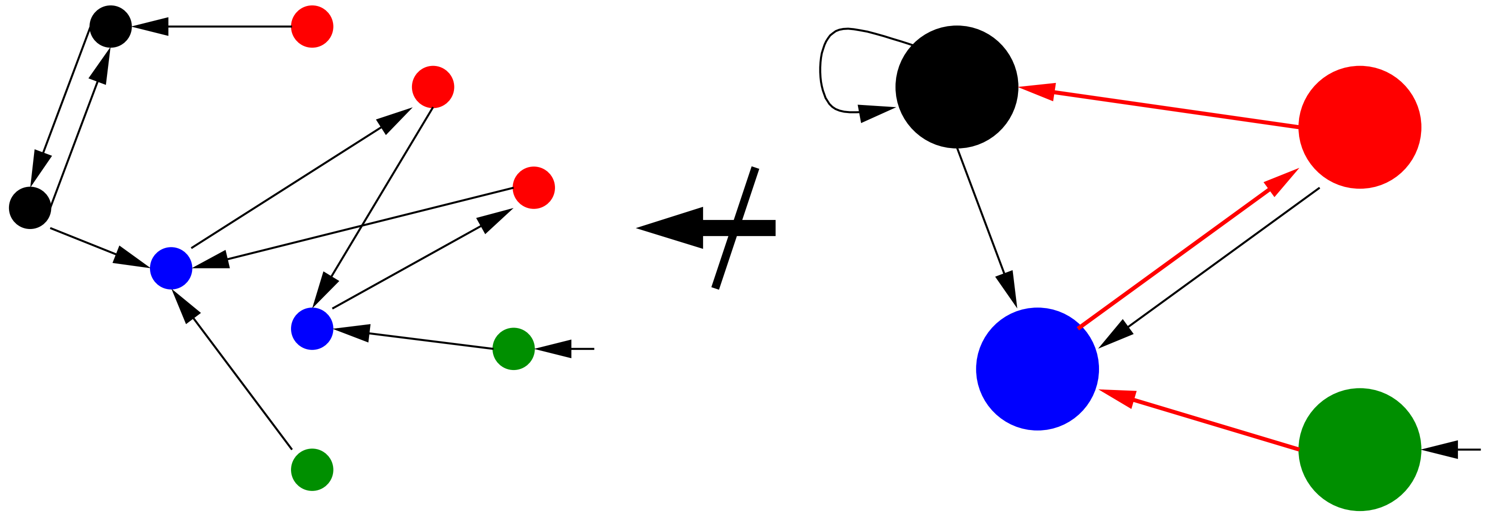
Abstraction gives rise to additional paths in the system:

Every concrete run has got a corresponding run in the abstraction . . .



Abstraction gives rise to additional paths in the system:

...but not every abstract run has got a corresponding run in the concrete system.



Suppose that $\mathcal{K}' \not\models \phi$, where ρ is a counterexample.

Check whether there is a run in $\mathcal{K}$ that "corresponds" to ρ.

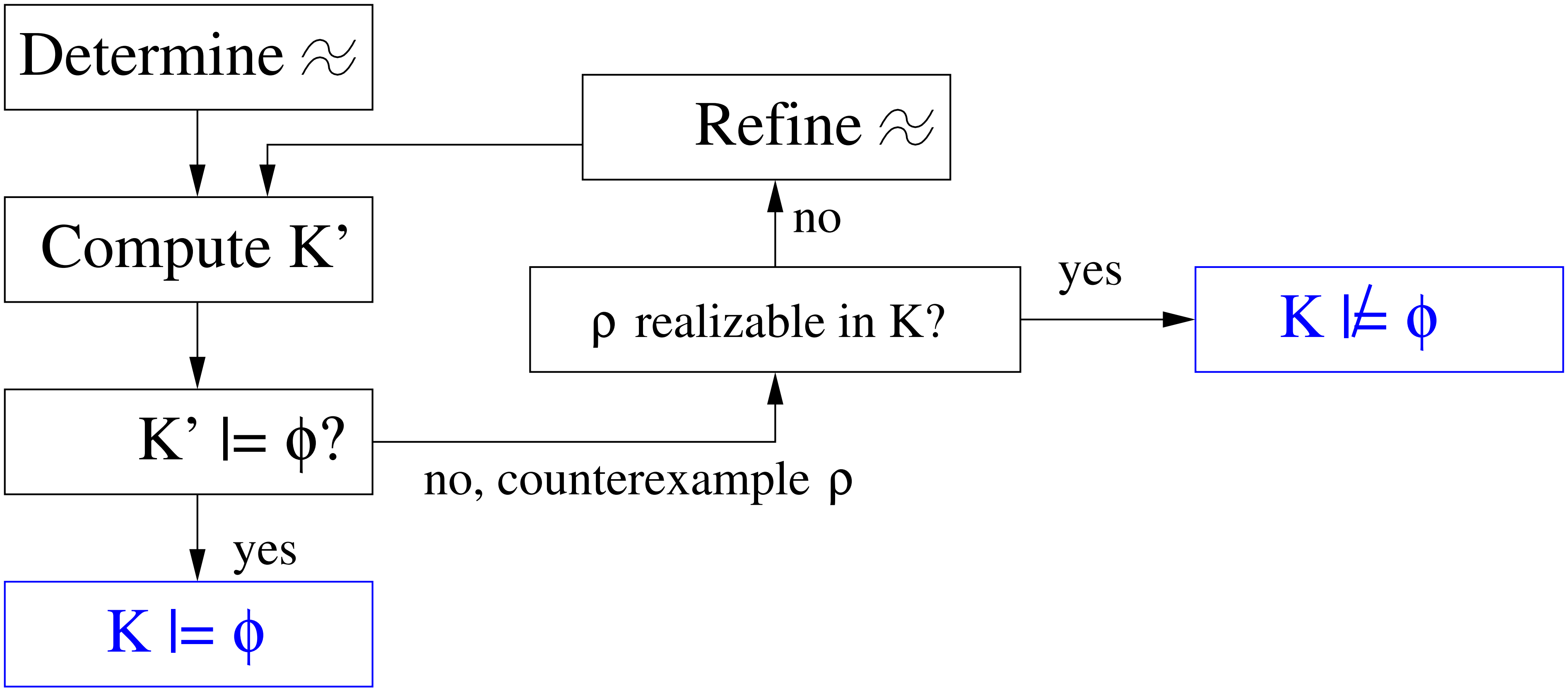
If yes, then $\mathcal{K} \not\models \phi$.

If no, then we can use ρ to refine the abstraction; i.e. we remove some equivalences from the relation H, introducing additional distinct states in $\mathcal{K}'$ so that ρ disappears.

The refinement can be repeated until a definite answer for $\mathcal{K} \models \phi$ (positive or negative) can be determined. This technique is called counterexample-guided abstraction refinement (CEGAR) [Clarke et al, 1994].

The abstraction-refinement cycle

Input: $\mathcal{K}, \phi$



Simulation of ρ

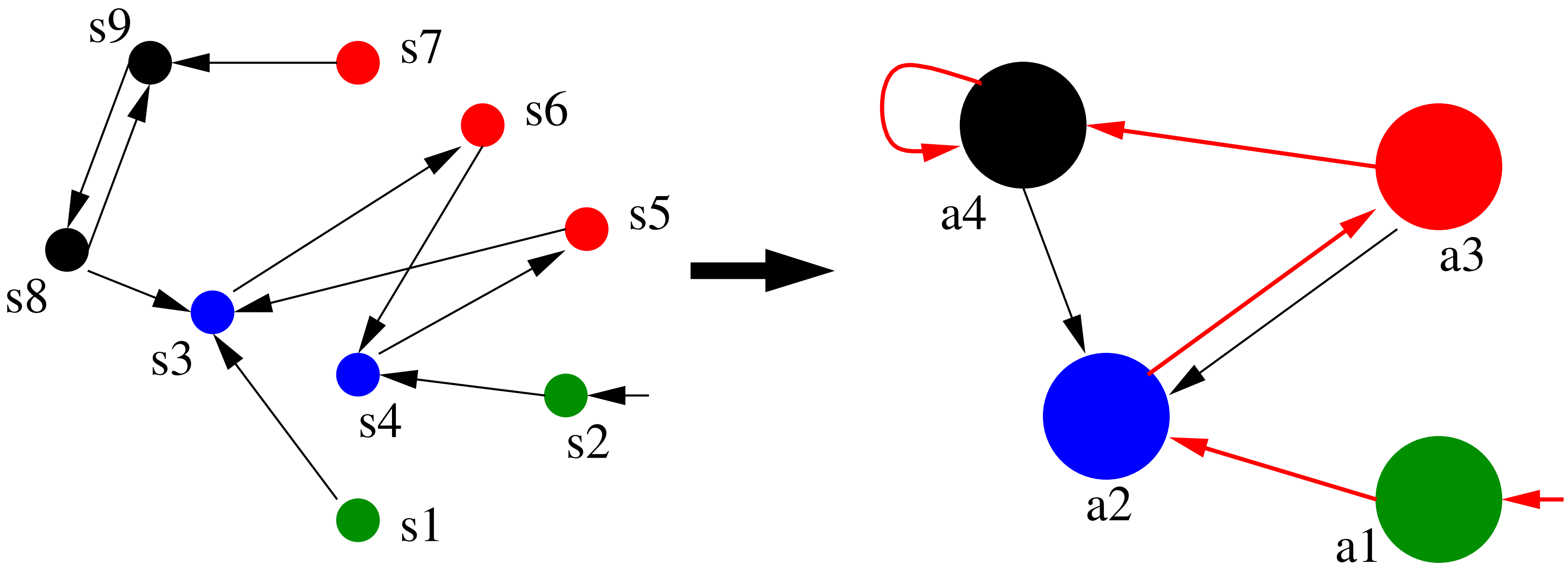
Problem: Given a counterexample ρ, is there a run corresponding to ρ in $\mathcal{K}$?

Solution: "Simulate" ρ on $\mathcal{K}$.

Remark: Any counterexample ρ can be partitioned into a finite stem and a finite loop, i.e. $\rho = w_S w_L^\omega$ for suitable w_S, w_L.

Case distinction: The simulation may fail in the stem or in the loop.

Example 1: $\mathbf{G}\,\neg$*black*



Abstraction yields a counterexample with stem $a_1 a_2 a_3 a_4$ and loop a_4.

Simulating the stem

Let $w_S = b_0 \cdots b_k$.

Start with $S_0 = \{r\}$. (We have $b_0 = [r]$.)

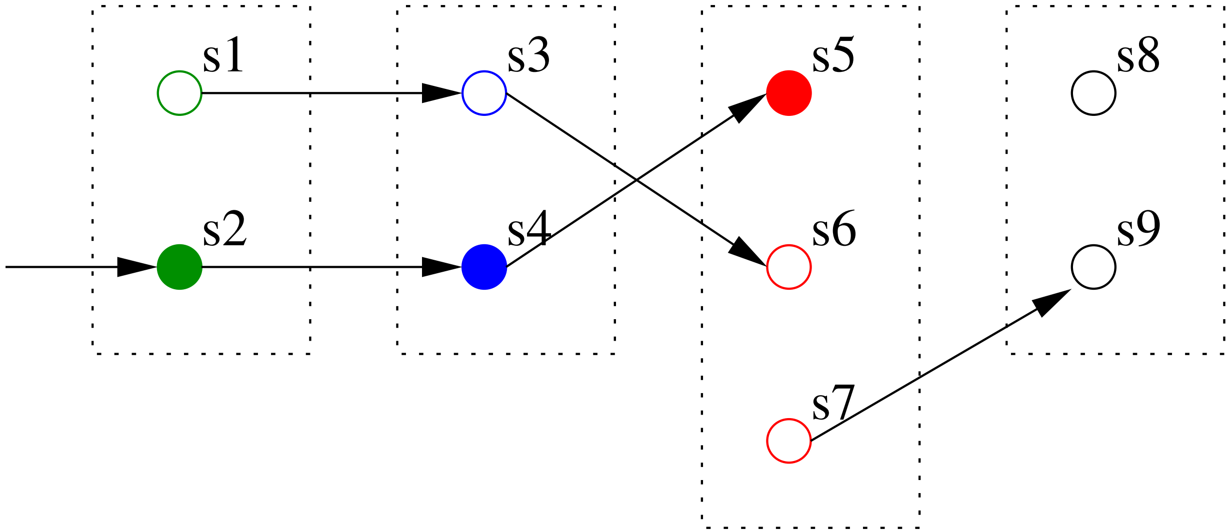
For $i = 1, \ldots, k$, compute $S_i = \{\, t \mid t \in b_i \wedge \exists s \in S_{i-1} : s \to t \,\}$.

If $S_k \neq \emptyset$, then there is a concrete correspondence for w_S.

If $S_k = \emptyset$: Find the smallest index ℓ with $S_\ell = \emptyset$: The refinement should distinguish the states in $S_{\ell-1}$ and those $b_{\ell-1}$-states that have immediate successors in b_ℓ.

Example: $w_S = a_1 a_2 a_3 a_4$

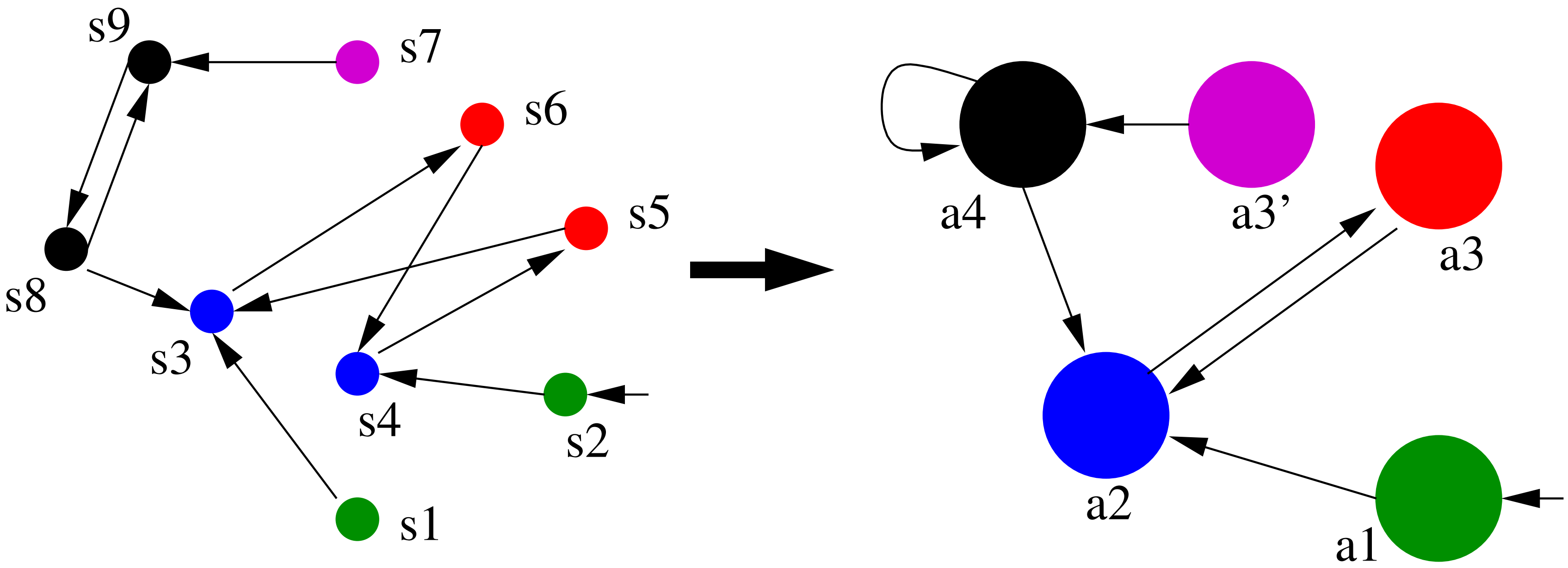
$S_0 = \{s_2\}, \qquad S_1 = \{s_4\}, \qquad S_2 = \{s_5\}, \qquad S_3 = \emptyset.$



In the next refinement, s_5 and s_7 must be distinguished.

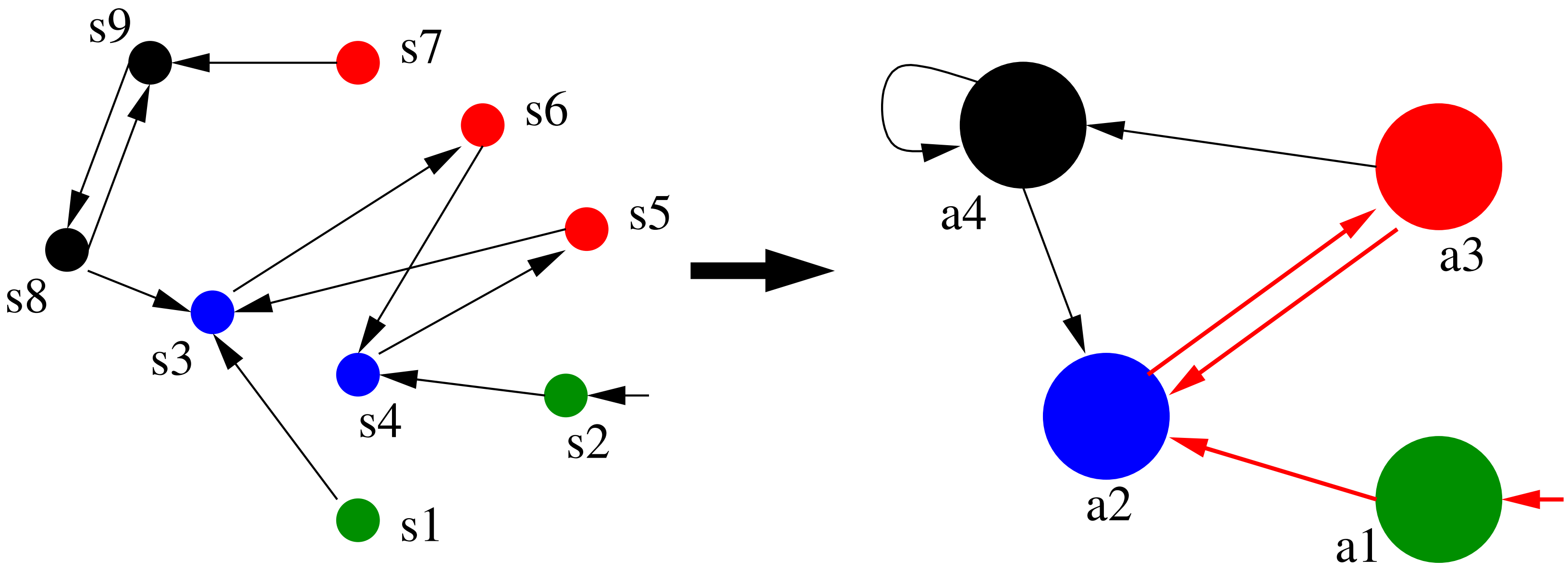
Possible new equivalence classes: $\{s_5, s_6\}, \{s_7\}$ or $\{s_5\}, \{s_6, s_7\}$.

Next try: $\mathbf{G}\,\neg$*black* with refinement



The new abstraction does not yield any counterexample; therefore, $\mathbf{G}\,\neg$*black* also holds in the concrete system.

Example 2: **F G** *red*



The abstraction yields a counterexample with stem $a_1 a_2$ and loop $a_3 a_2$.

Simulating a loop

Assume $w_S = b_0 \cdots b_k, \quad w_L = c_1 \cdots c_\ell$

w_S is simulated as before, however w_L may have to be simulated multiple times.

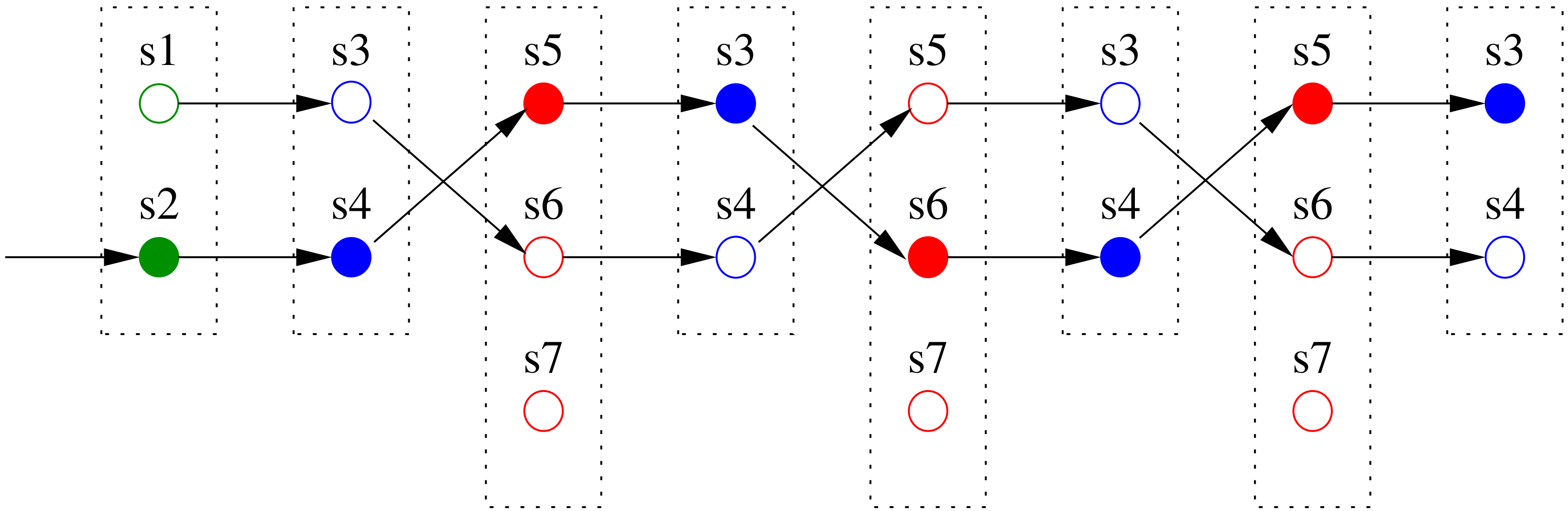
Let m be the size of the smallest equivalence class in w_L:

$$m = \min_{i=1,\dots,\ell} |c_i|$$

Then we simulate the path $w_S w_L^{m+1}$; doing so, either the simulation will fail, or we will discover a real counterexample.

Refinement: same as before.

Example: $w_S = a_1a_2$, $w_L = a_3a_2$, $m = 2$



The simulation succeeds because there is a loop around s_4.

Thus, there is a real counterexample, so $\mathcal{K} \not\models \phi$.