

Strategy Extraction in Reachability Games

Niklas Een, Alexander Legg, Nina Narodytska and Leonid Ryzhyk



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Outline

- Motivation
- Introduction into reachability games
- CEGAR-based algorithm (EvaSolver)
- Strategy extraction
- Conclusion

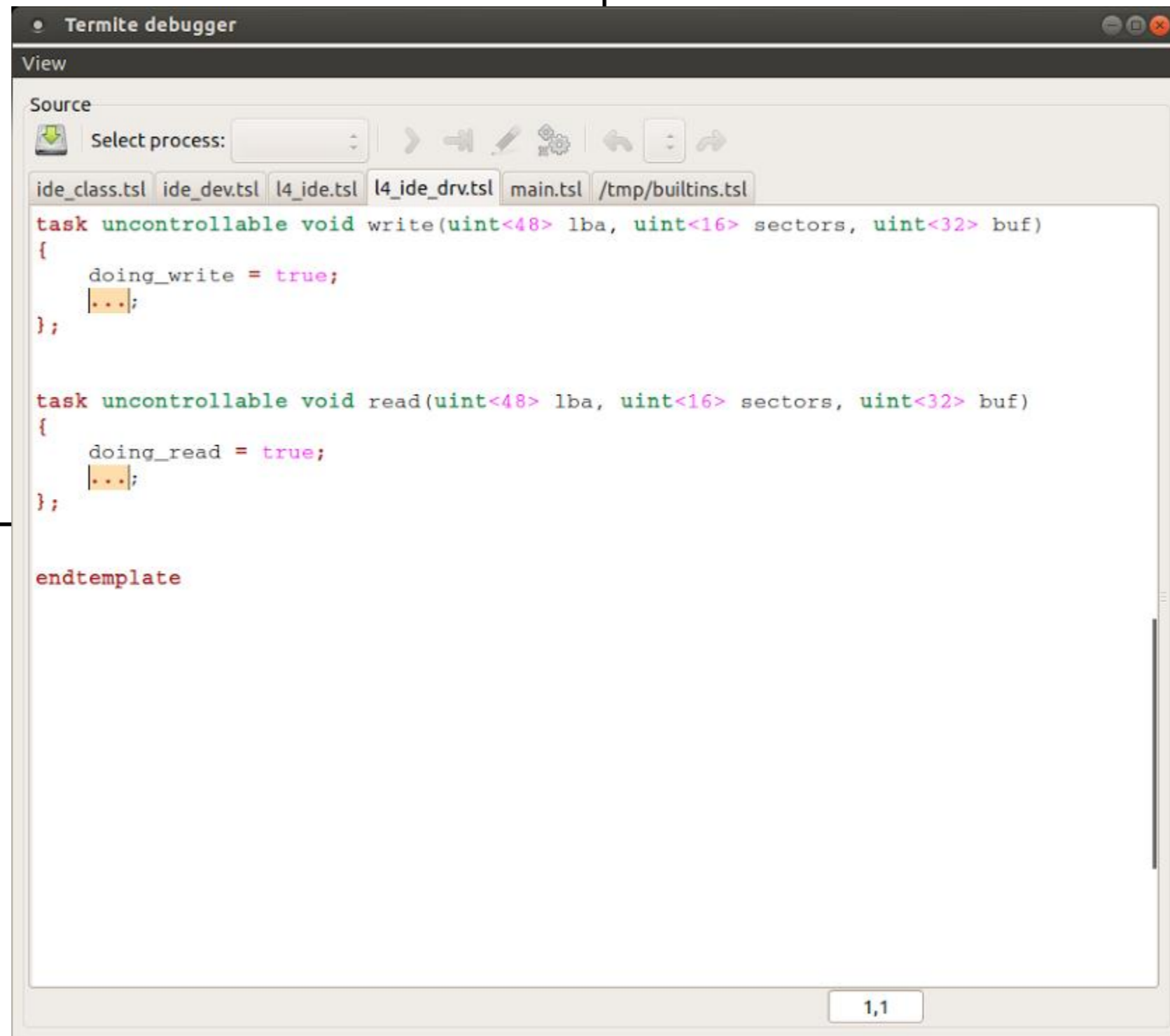


Automatic Driver Synthesis Project (funded by Intel)



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Practical Device Driver Synthesis (OSDI'14)



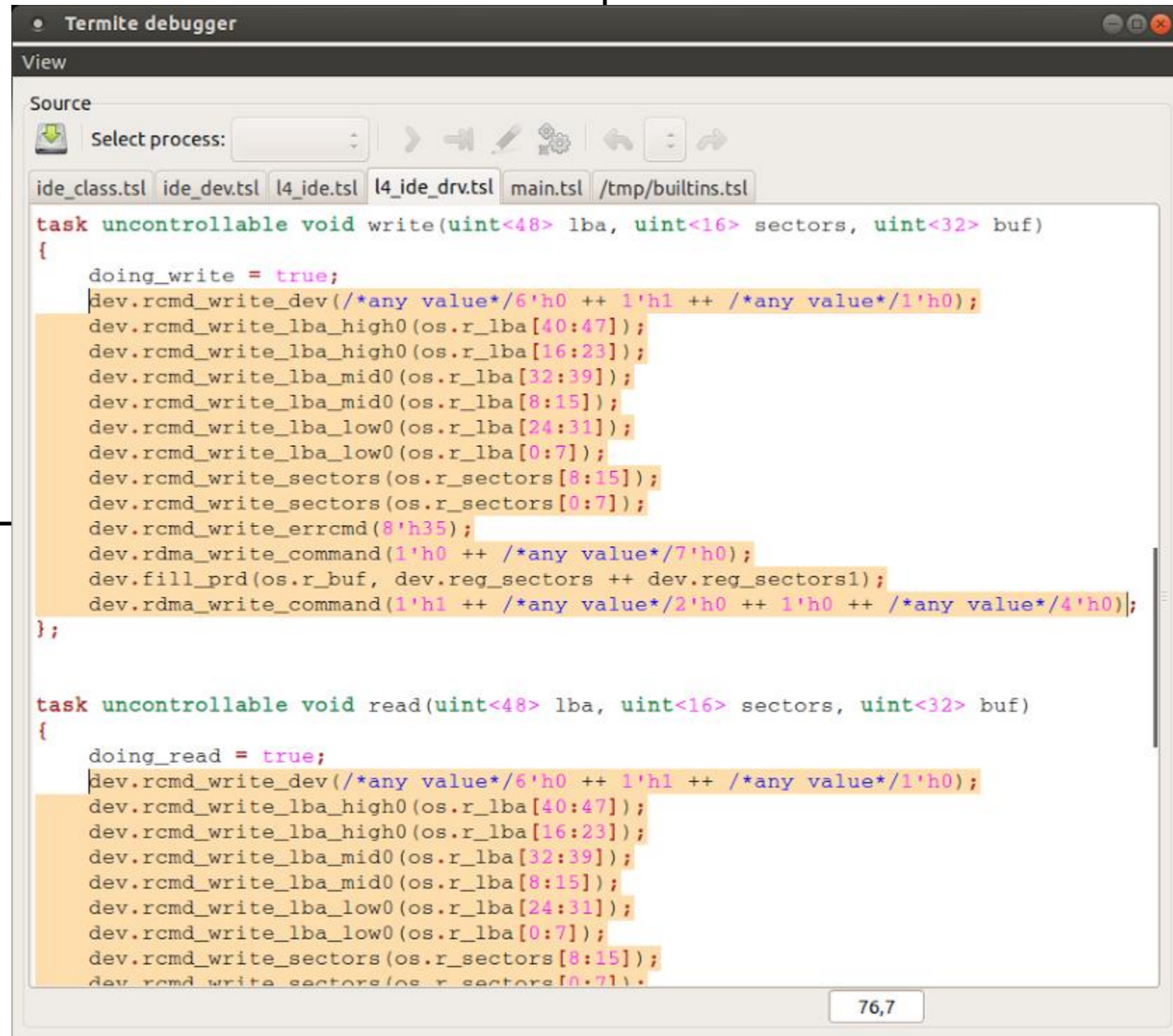
```
Termite debugger
View
Source
Select process:
ide_class.tsl ide_dev.tsl l4_ide.tsl l4_ide_drv.tsl main.tsl /tmp/builtins.tsl
task uncontrollable void write(uint<48> lba, uint<16> sectors, uint<32> buf)
{
    doing_write = true;
    ...;
};

task uncontrollable void read(uint<48> lba, uint<16> sectors, uint<32> buf)
{
    doing_read = true;
    ...;
};

endtemplate
1,1
```



Practical Device Driver Synthesis (OSDI'14)



The screenshot shows the Termitte debugger interface. The 'View' pane displays the source code for two tasks: 'write' and 'read'. The code is written in a language that uses bit-level operations and device-specific functions. The 'write' task is marked as 'uncontrollable' and includes a 'doing_write = true;' flag. It performs several device register writes for lba_high0, lba_mid0, lba_low0, sectors, and errcmd, followed by rdma_write_command calls. The 'read' task is also marked as 'uncontrollable' and includes a 'doing_read = true;' flag. It performs similar device register writes for lba_high0, lba_mid0, lba_low0, and sectors. The code is highlighted in yellow in the original image.

```
task uncontrollable void write(uint<48> lba, uint<16> sectors, uint<32> buf)
{
    doing_write = true;
    dev.rcmd_write_dev(/*any value*/6'h0 ++ 1'h1 ++ /*any value*/1'h0);
    dev.rcmd_write_lba_high0(os.r_lba[40:47]);
    dev.rcmd_write_lba_high0(os.r_lba[16:23]);
    dev.rcmd_write_lba_mid0(os.r_lba[32:39]);
    dev.rcmd_write_lba_mid0(os.r_lba[8:15]);
    dev.rcmd_write_lba_low0(os.r_lba[24:31]);
    dev.rcmd_write_lba_low0(os.r_lba[0:7]);
    dev.rcmd_write_sectors(os.r_sectors[8:15]);
    dev.rcmd_write_sectors(os.r_sectors[0:7]);
    dev.rcmd_write_errcmd(8'h35);
    dev.rdma_write_command(1'h0 ++ /*any value*/7'h0);
    dev.fill_prd(os.r_buf, dev.reg_sectors ++ dev.reg_sectors1);
    dev.rdma_write_command(1'h1 ++ /*any value*/2'h0 ++ 1'h0 ++ /*any value*/4'h0);
};

task uncontrollable void read(uint<48> lba, uint<16> sectors, uint<32> buf)
{
    doing_read = true;
    dev.rcmd_write_dev(/*any value*/6'h0 ++ 1'h1 ++ /*any value*/1'h0);
    dev.rcmd_write_lba_high0(os.r_lba[40:47]);
    dev.rcmd_write_lba_high0(os.r_lba[16:23]);
    dev.rcmd_write_lba_mid0(os.r_lba[32:39]);
    dev.rcmd_write_lba_mid0(os.r_lba[8:15]);
    dev.rcmd_write_lba_low0(os.r_lba[24:31]);
    dev.rcmd_write_lba_low0(os.r_lba[0:7]);
    dev.rcmd_write_sectors(os.r_sectors[8:15]);
    dev.rcmd_write_sectors(os.r_sectors[0:7]);
};
```

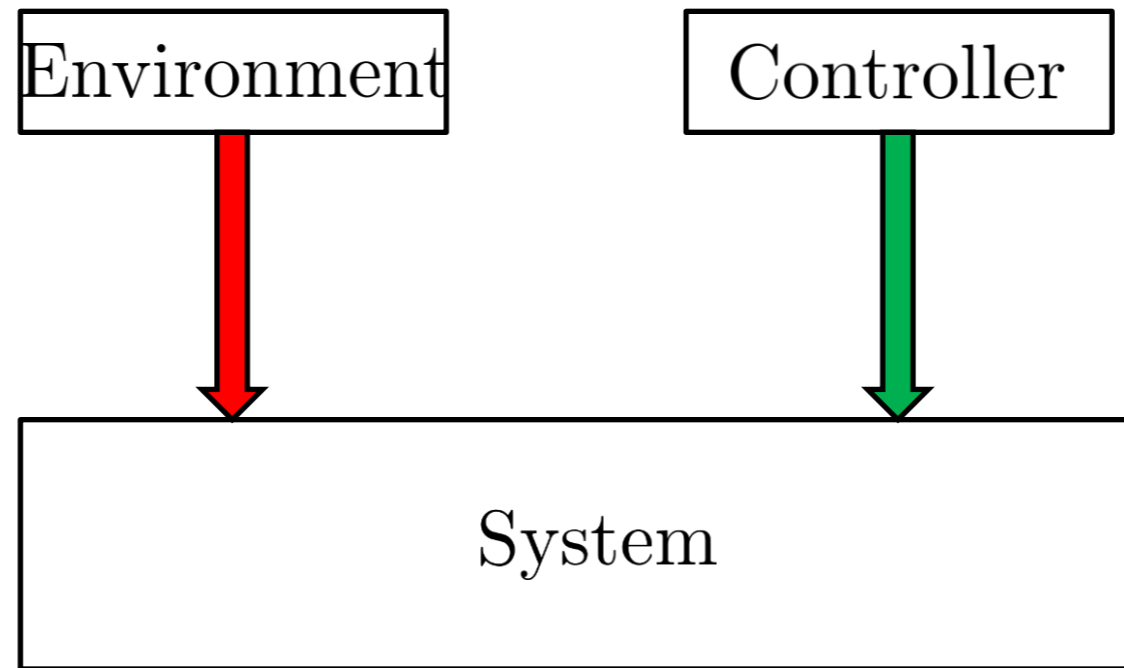


Introduction into reachability games



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Reachability Games



Can the system be controlled
to satisfy ϕ ?



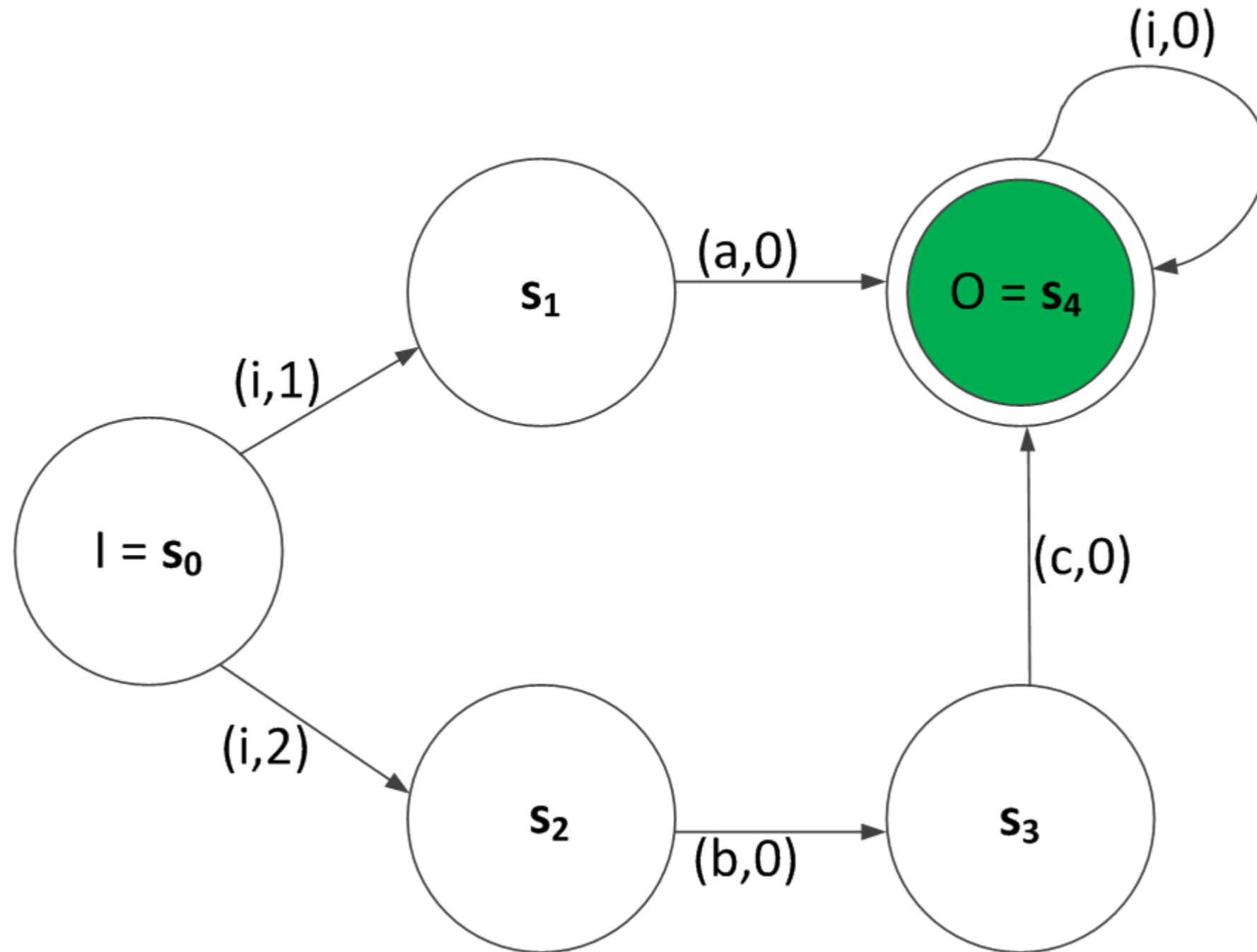
Reachability games

A reachability game $G = \langle S, L_c, L_u, I, O, \delta \rangle$ consists of

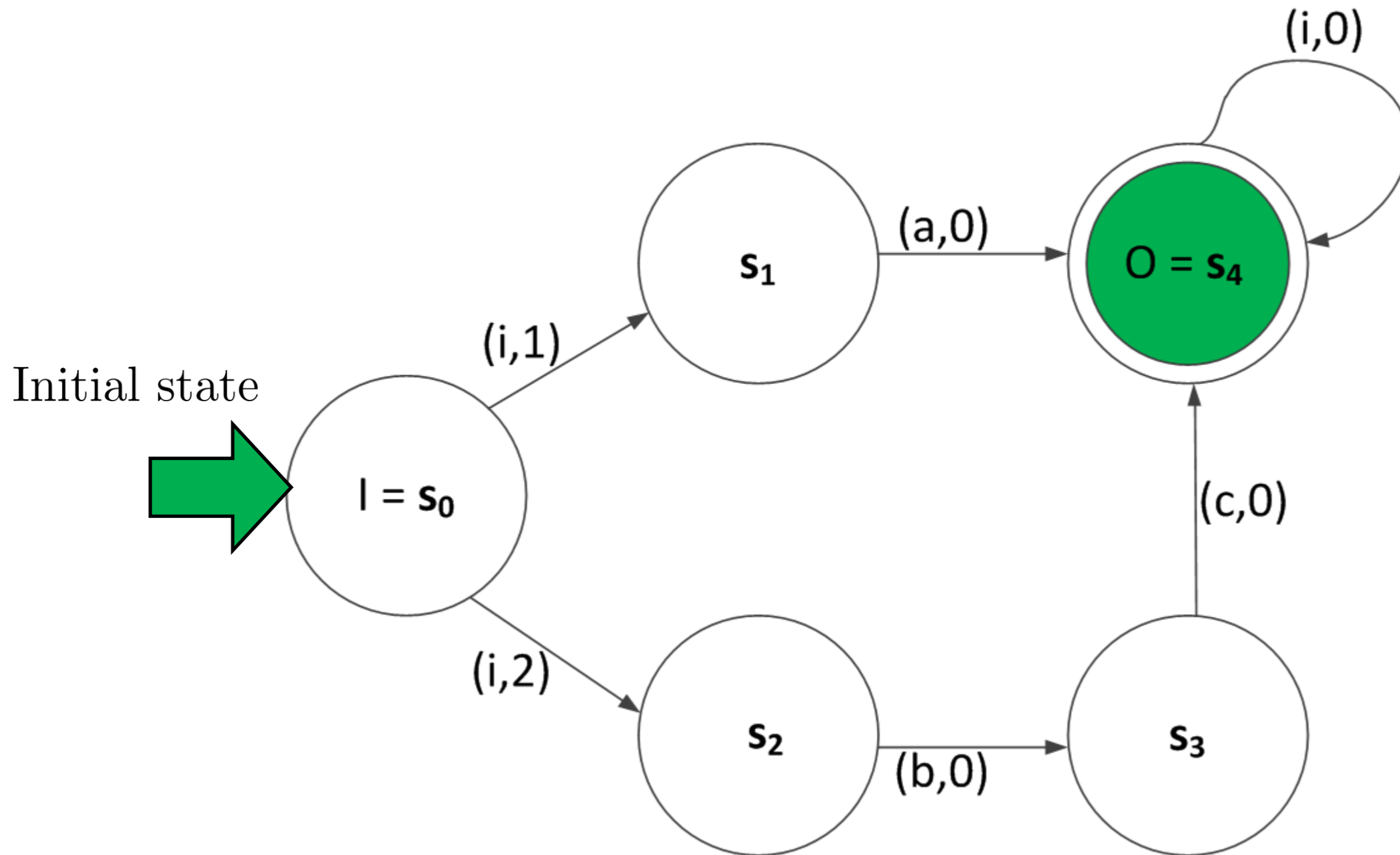
- a set of states S ,
- controllable actions L_c ,
- uncontrollable actions L_u ,
- initial state $I \in S$,
- a set $O \in 2^S$ of goal states, and
- a transition function $\delta : (S, L_c, L_u) \rightarrow S$.



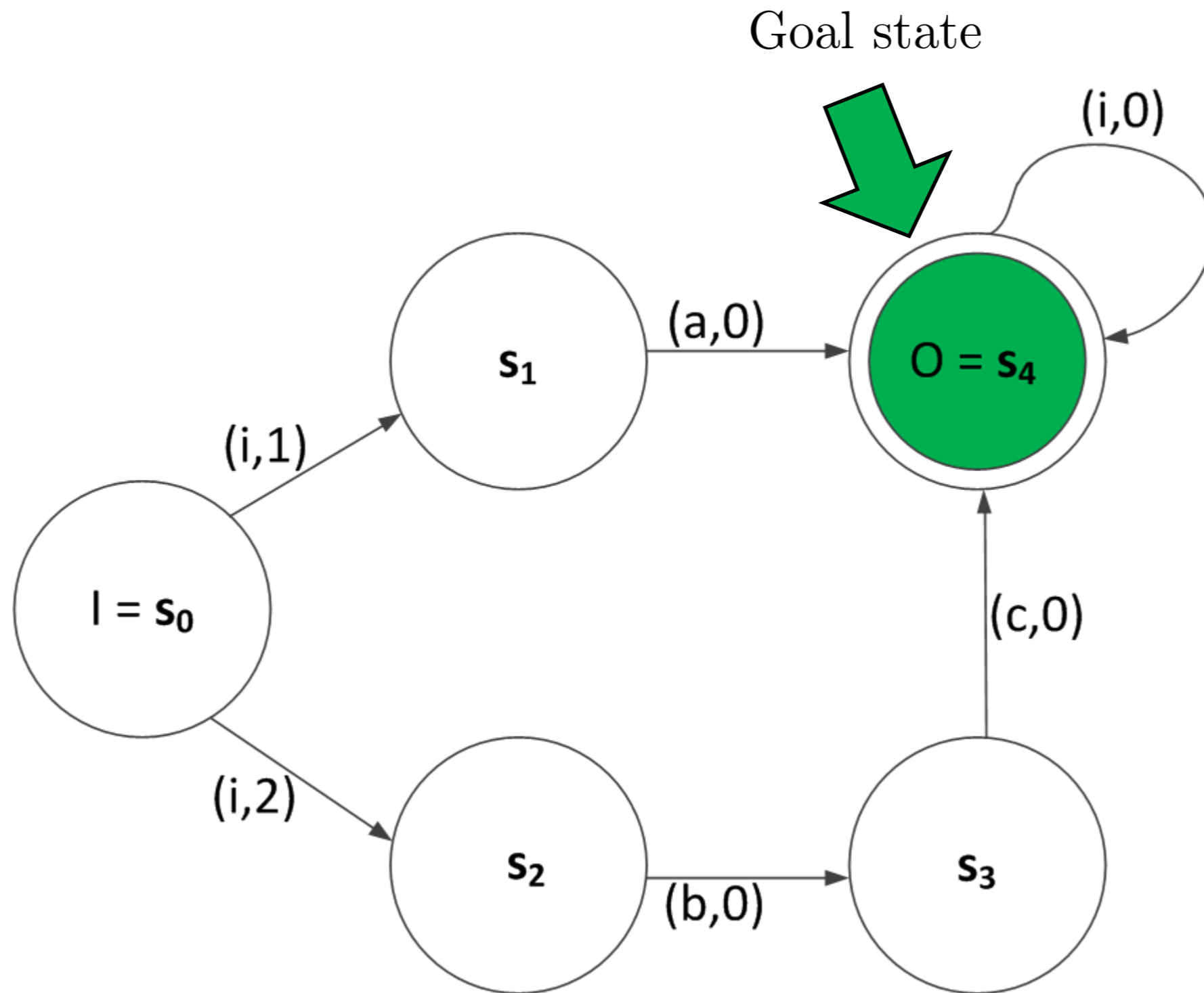
Reachability games



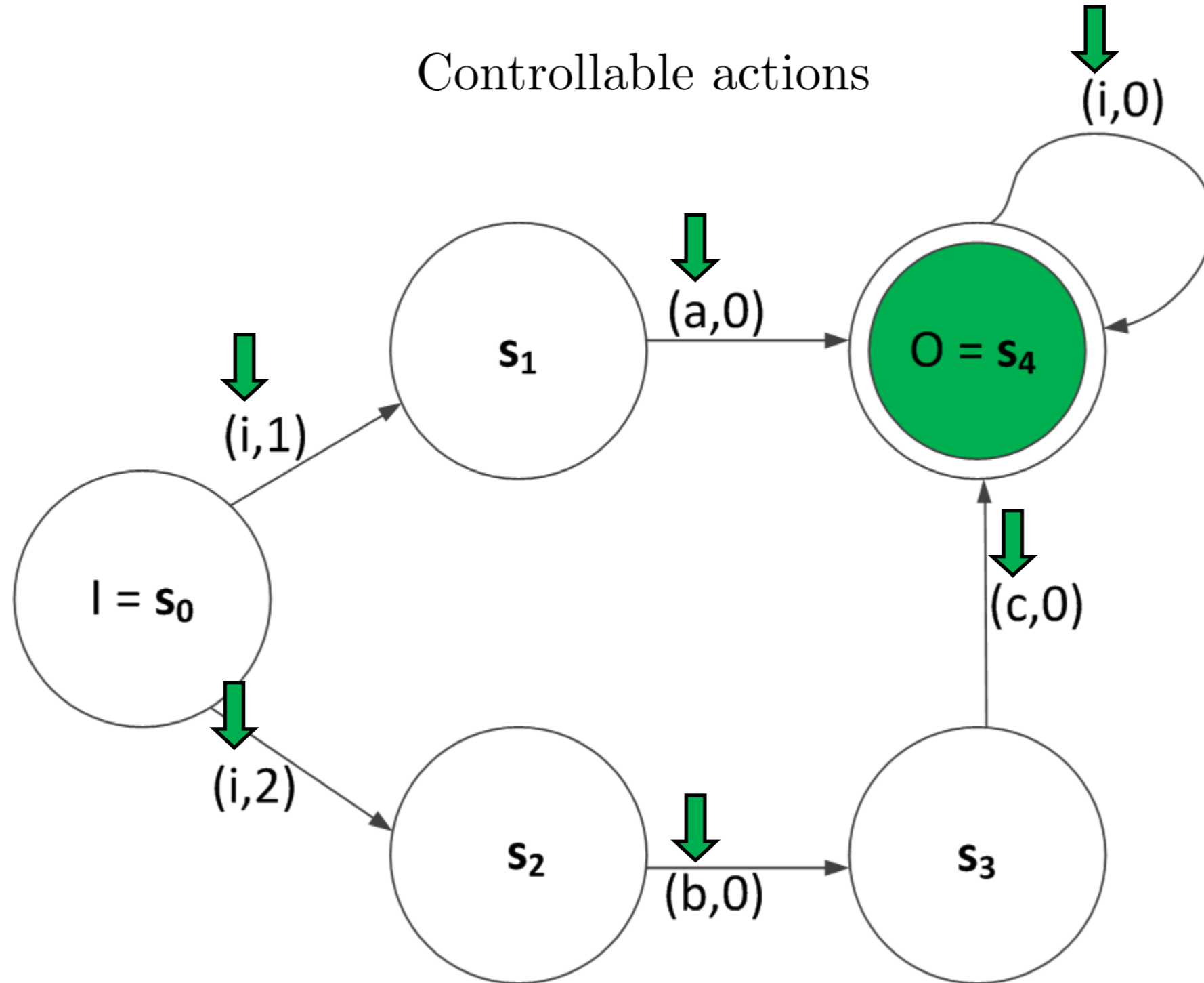
Reachability games



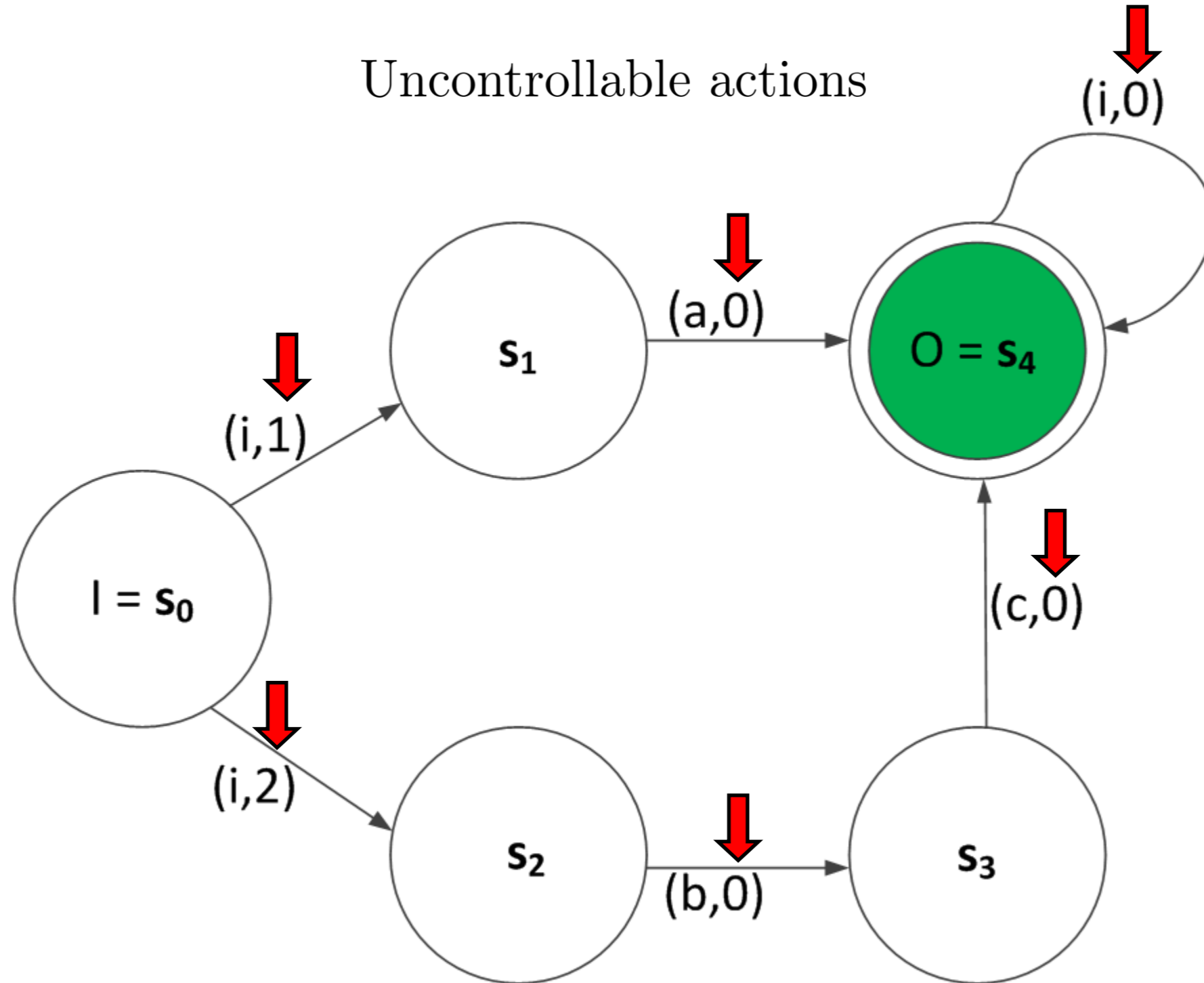
Reachability games



Reachability games



Reachability games



Reachability games

The goal of the controller is to reach the goal state regardless of the behaviour of the environment.



Reachability games

The goal of the controller is to reach the goal state
in n steps
regardless of the behaviour of the environment.



Reachability games

A controller strategy $\pi : S \rightarrow L_c$

associates with every state a controllable action to play in this state.



Reachability games

Given a bound n , π is a *winning* strategy in state \mathbf{s} at round $i \leq n$ if
any sequence $(\mathbf{s}_i, \mathbf{u}_i, \mathbf{s}_{i+1}, \mathbf{u}_{i+1}, \dots, \mathbf{s}_n)$,
such that $\mathbf{s}_i = \mathbf{s}$ and $\mathbf{s}_{k+1} = \delta(\mathbf{s}_k, \pi(\mathbf{s}_k), \mathbf{u}_k)$,
visits the goal set: $\exists j \in [i, n]. \mathbf{s}_j \in O$.



Winning strategy: Existence vs Extraction



Reachability games solver



Reachability games solver



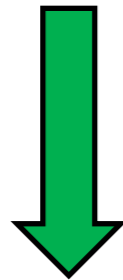
No



Reachability games solver



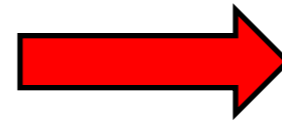
No



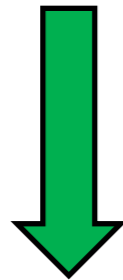
Yes, here is a strategy.



Reachability games solver



No

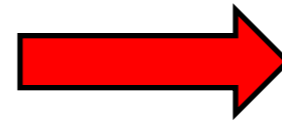


Yes, here is a strategy.

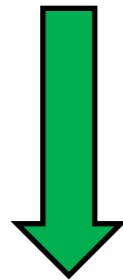
1. SAT-Based Synthesis Methods for Safety Specs, VMCAI'14
Roderick Bloem, Robert Knighofer, Martina Seidl
2. Solving Games Using Incremental Induction, IFM'13
Andreas Morgenstern, Manuel Gesell, Klaus Schneider



Reachability games solver



No



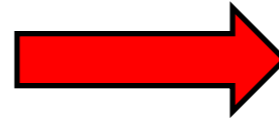
~~Yes, here is a strategy.~~

1. SAT-Based Synthesis Methods for Safety Specs, VMCAI'14
Roderick Bloem, Robert Knighofer, Martina Seidl

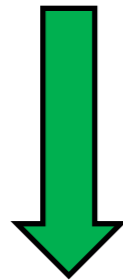
2. Solving Games Using Incremental Induction, IFM'13
Andreas Morgenstern, Manuel Gesell, Klaus Schneider



Reachability games solver



No



~~Yes, here is a strategy.~~

Yes, there is a strategy.

1. SAT-Based Synthesis Methods for Safety Specs, VMCAI'14
Roderick Bloem, Robert Knighofer, Martina Seidl

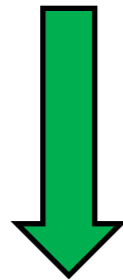
2. Solving Games Using Incremental Induction, IFM'13
Andreas Morgenstern, Manuel Gesell, Klaus Schneider



EvaSolver
[Narodytska et al., CAV'14]



No



~~Yes, here is a strategy.~~

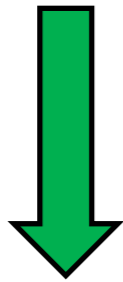
Yes, there is a strategy.



EvaSolver
[Narodytska et al., CAV'14]



No



~~Yes, here is a strategy.~~

~~Yes, there is a strategy.~~

Yes, here is a strategy certificate.



Game tree



Game tree



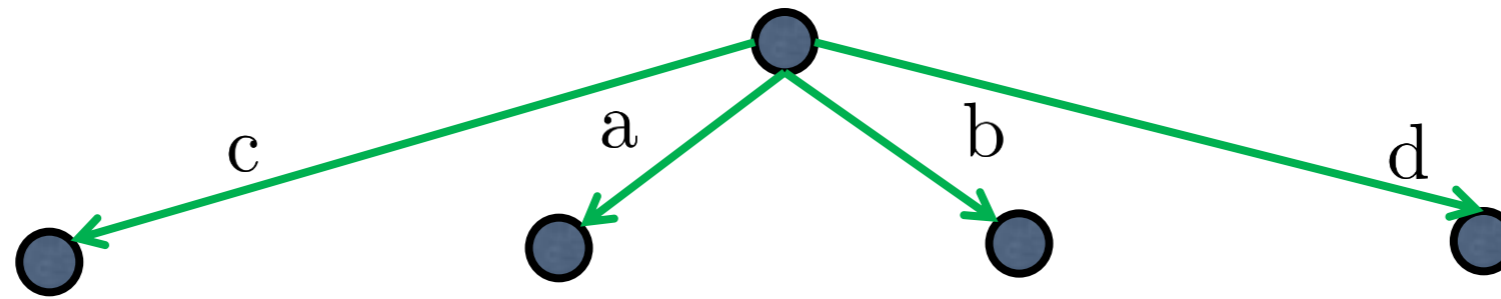
- controllable move
- uncontrollable move

Simplifying assumptions:

- consider reachability games
- bound the number of rounds in the game
- players strictly alternate



Game tree



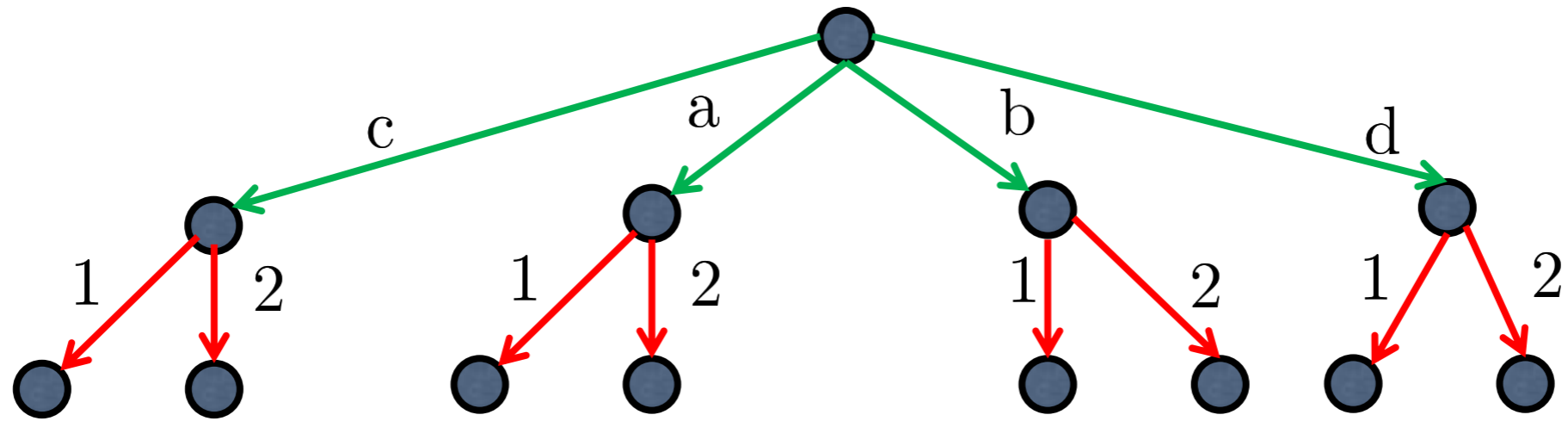
- controllable move
- uncontrollable move

Simplifying assumptions:

- consider reachability games
- bound the number of rounds in the game
- players strictly alternate



Game tree



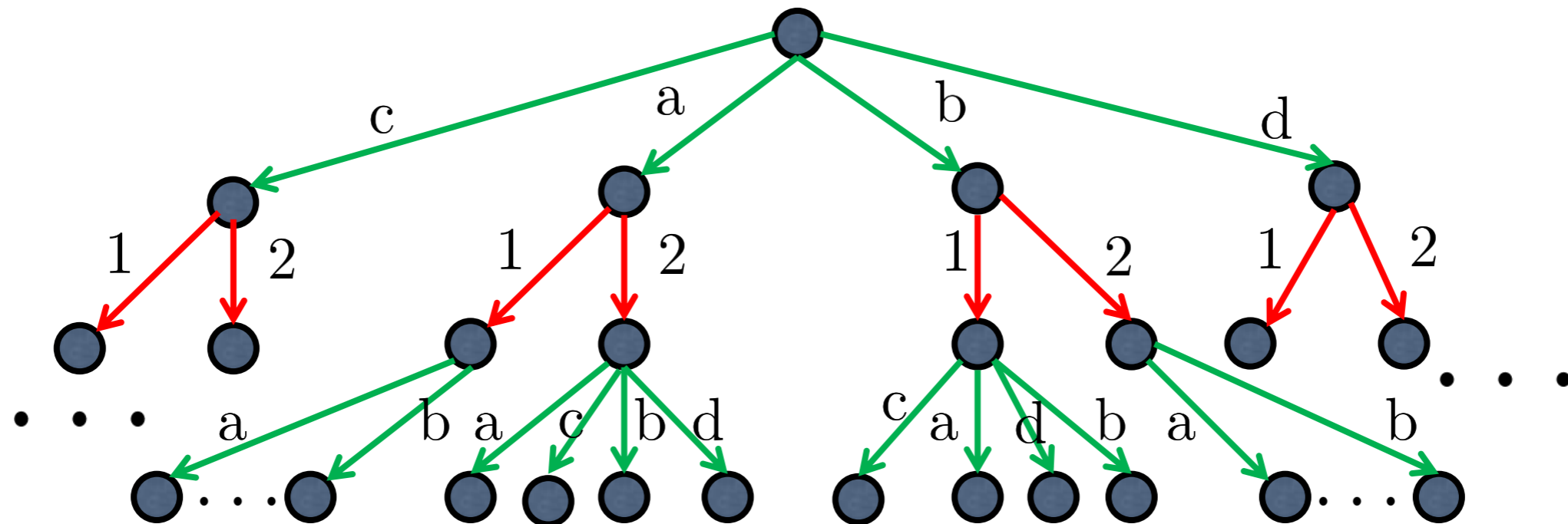
- controllable move
- uncontrollable move

Simplifying assumptions:

- consider reachability games
- bound the number of rounds in the game
- players strictly alternate



Game tree



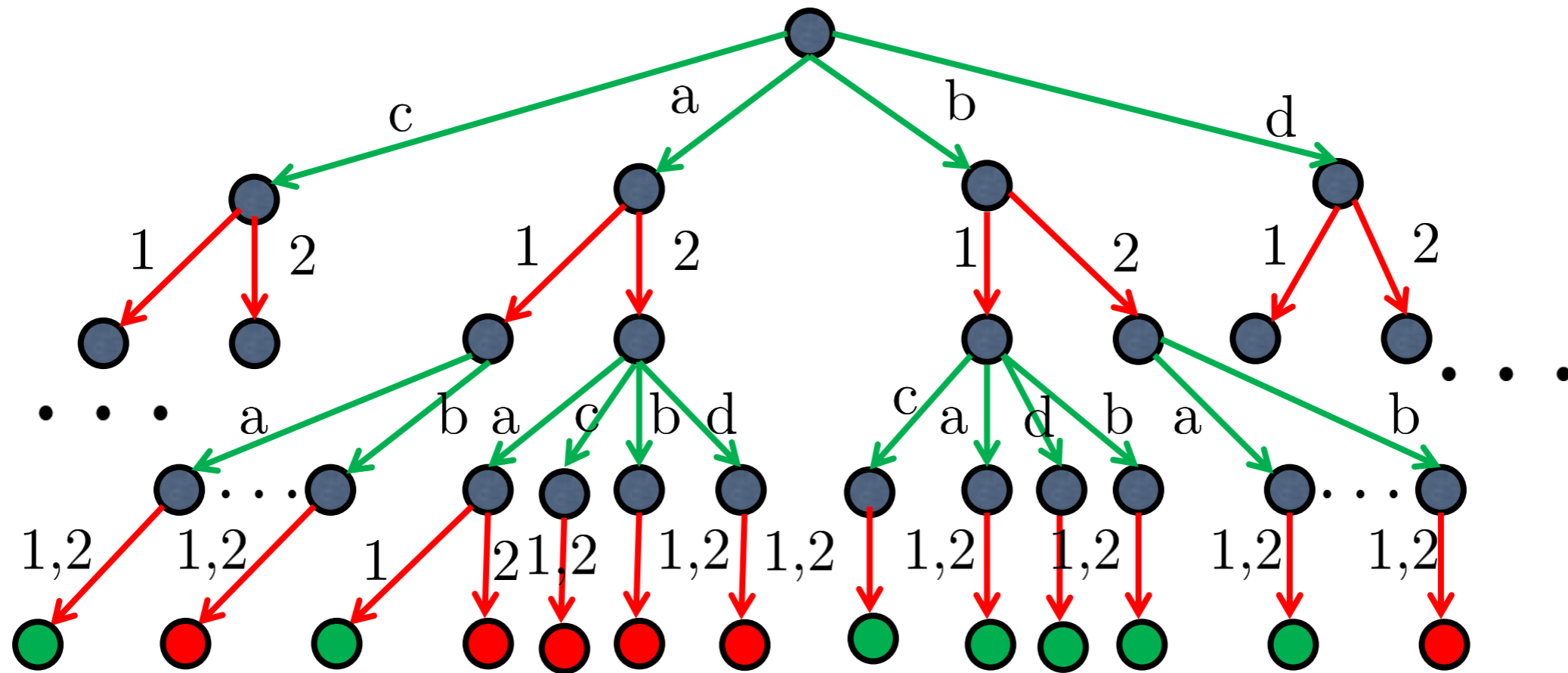
- controllable move
- uncontrollable move





Simplifying assumptions:

- consider reachability games
- bound the number of rounds in the game
- players strictly alternate



Game tree



-  controllable move
-  uncontrollable move
-  winning state
-  losing state

Simplifying assumptions:

- consider reachability games
- bound the number of rounds in the game
- players strictly alternate

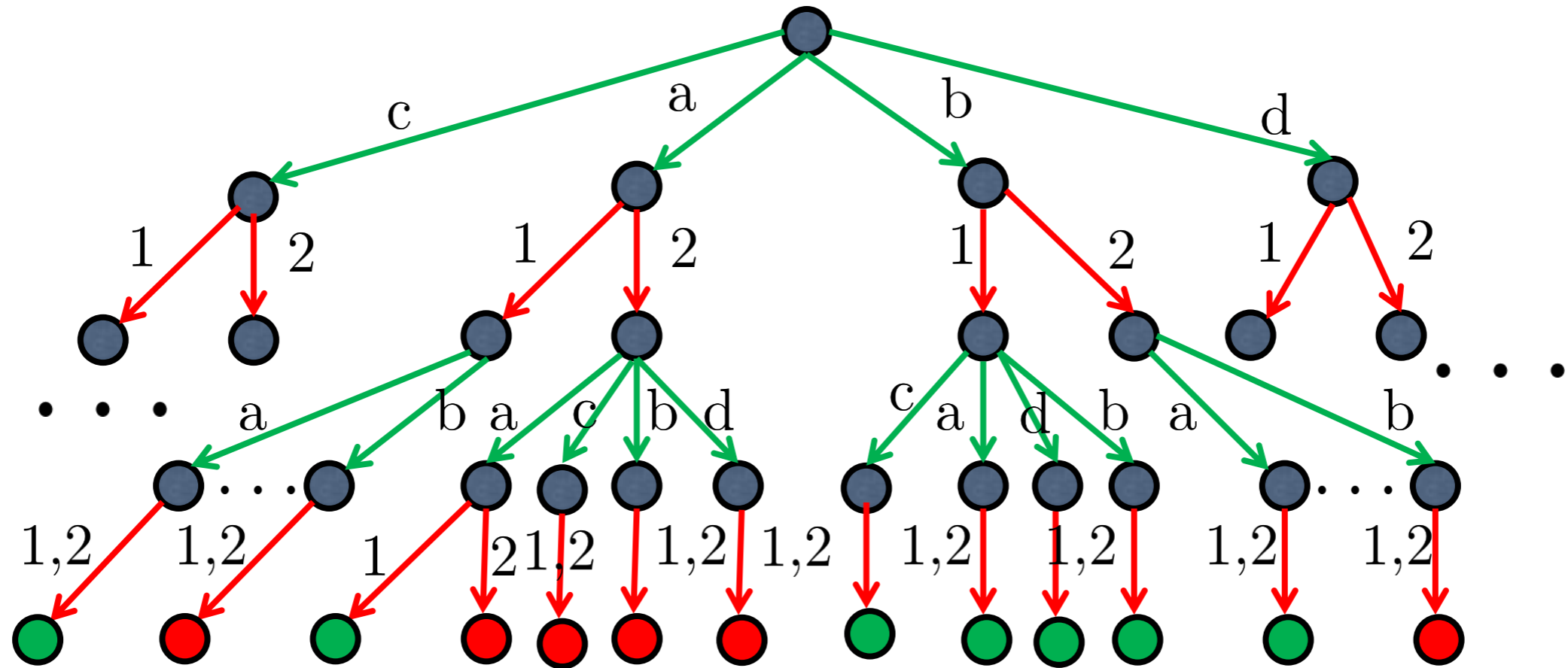






Winning strategy certificate



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Game tree



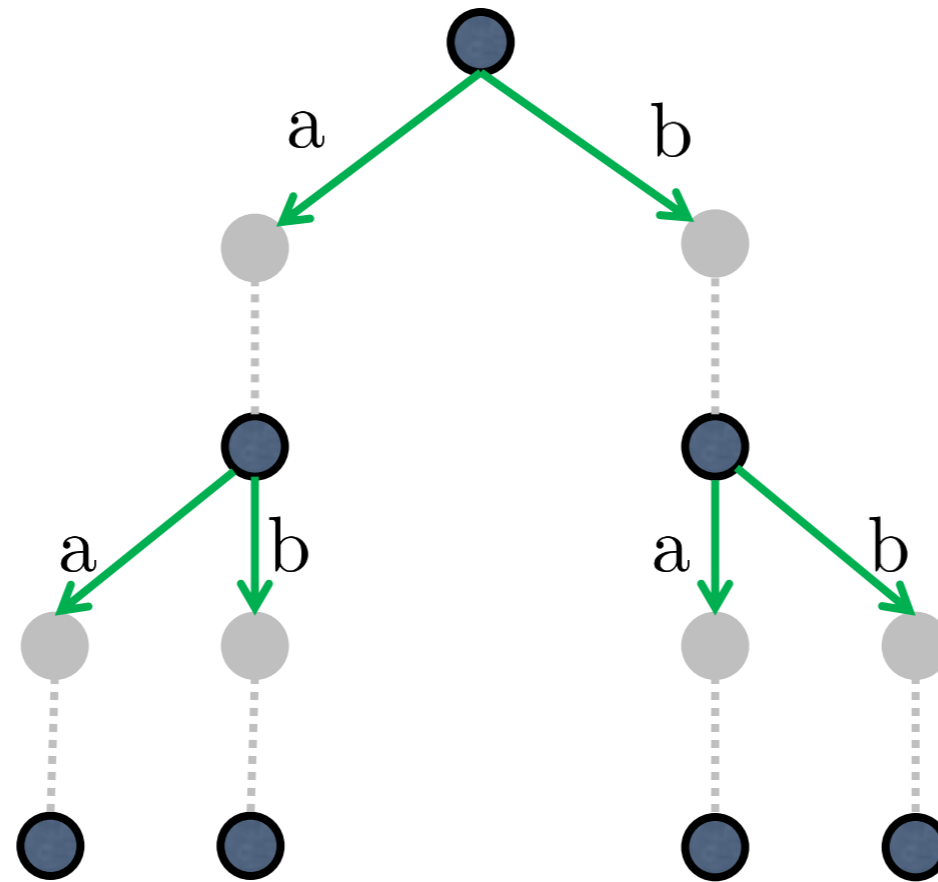
-  controllable move
-  uncontrollable move
-  winning state
-  losing state

Simplifying assumptions:

- consider reachability games
- bound the number of rounds in the game
- players strictly alternate



Abstract Game tree



→ controllable move

..... some uncontrollable move

restricts controllable player moves

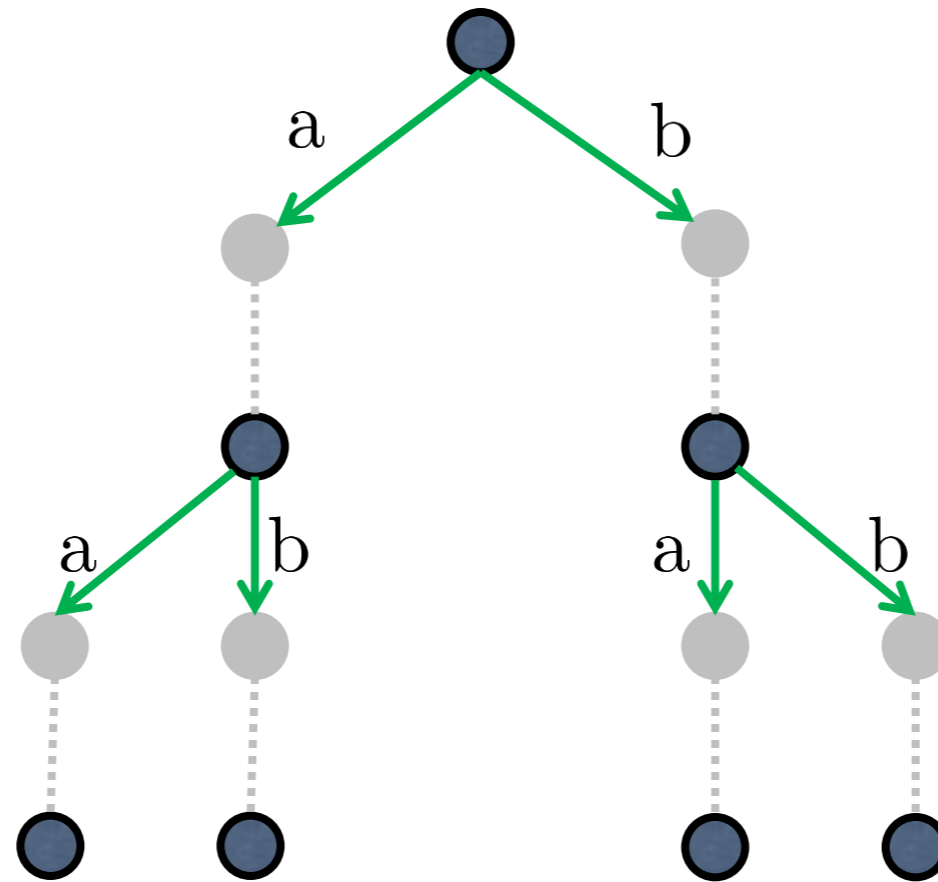


Verifying a certificate



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

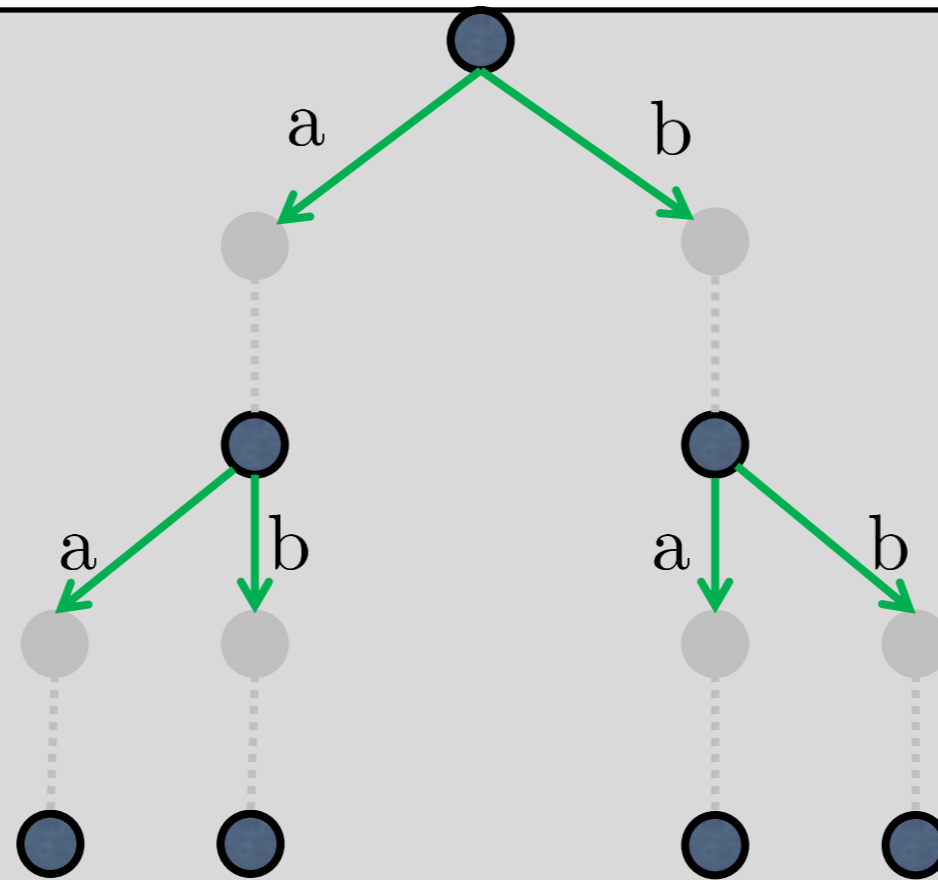
Abstract Game tree verification



Can the environment win a safety game if the controller moves are restricted to the ones in the tree?



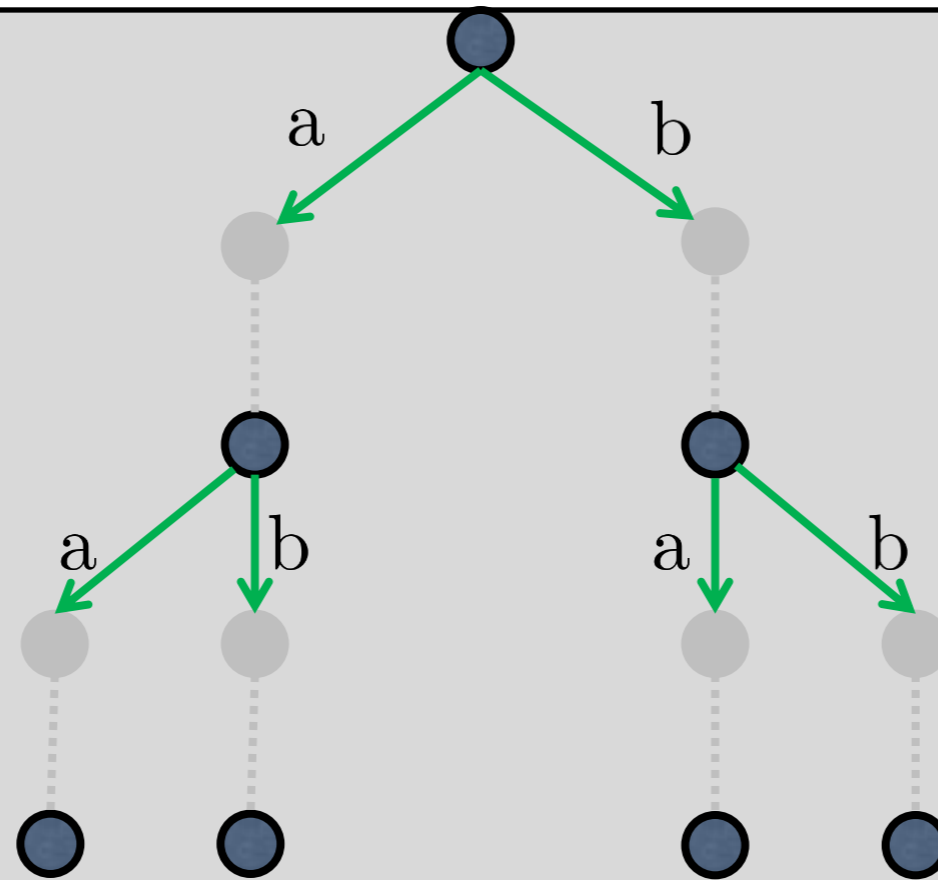
Abstract Game tree verification



Can the environment win a safety game if the controller moves are restricted to the ones in the tree?



Abstract Game tree verification

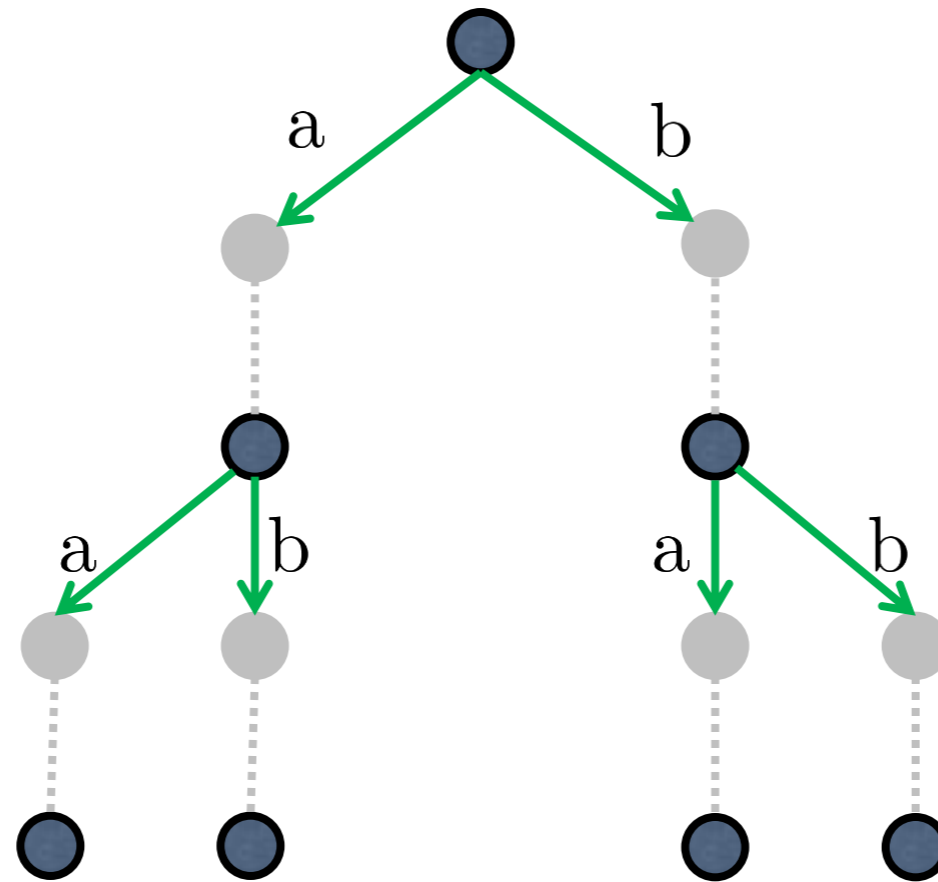


Can the environment win a safety game if the controller moves are restricted to the ones in the tree?

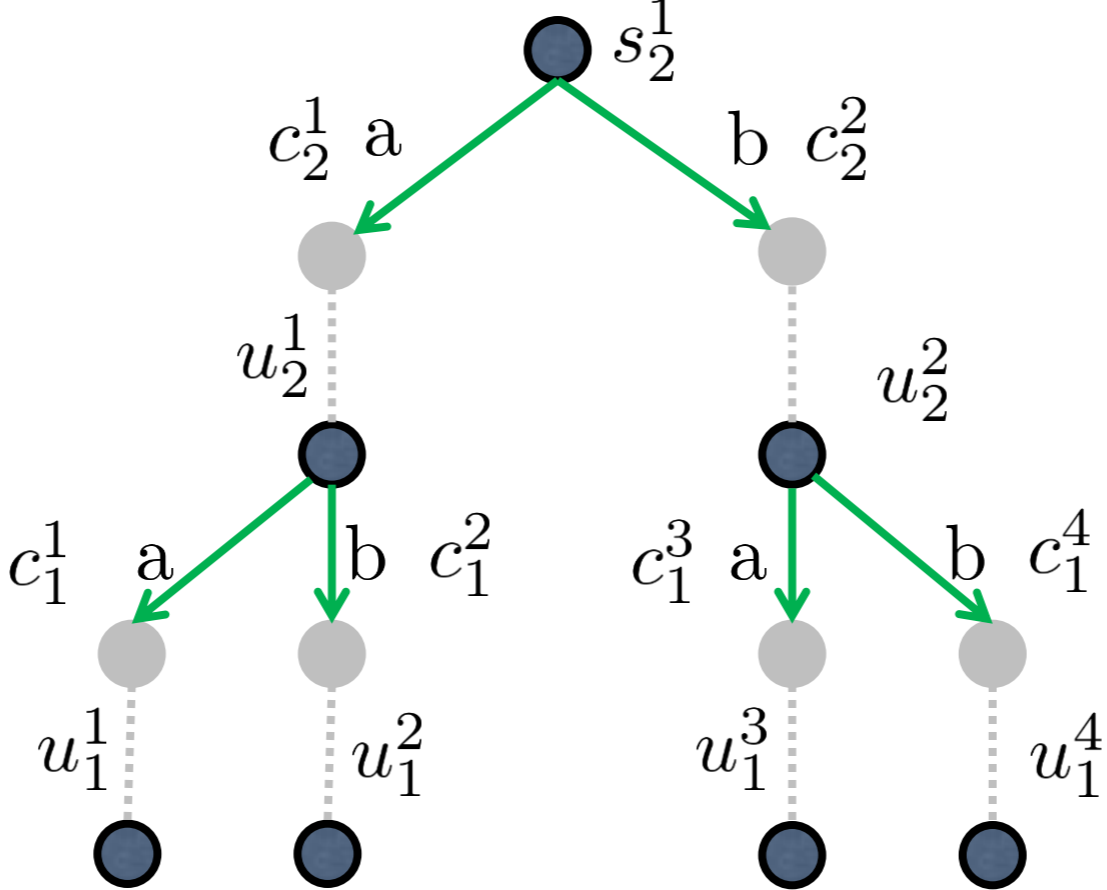
Encode into a SAT formula.



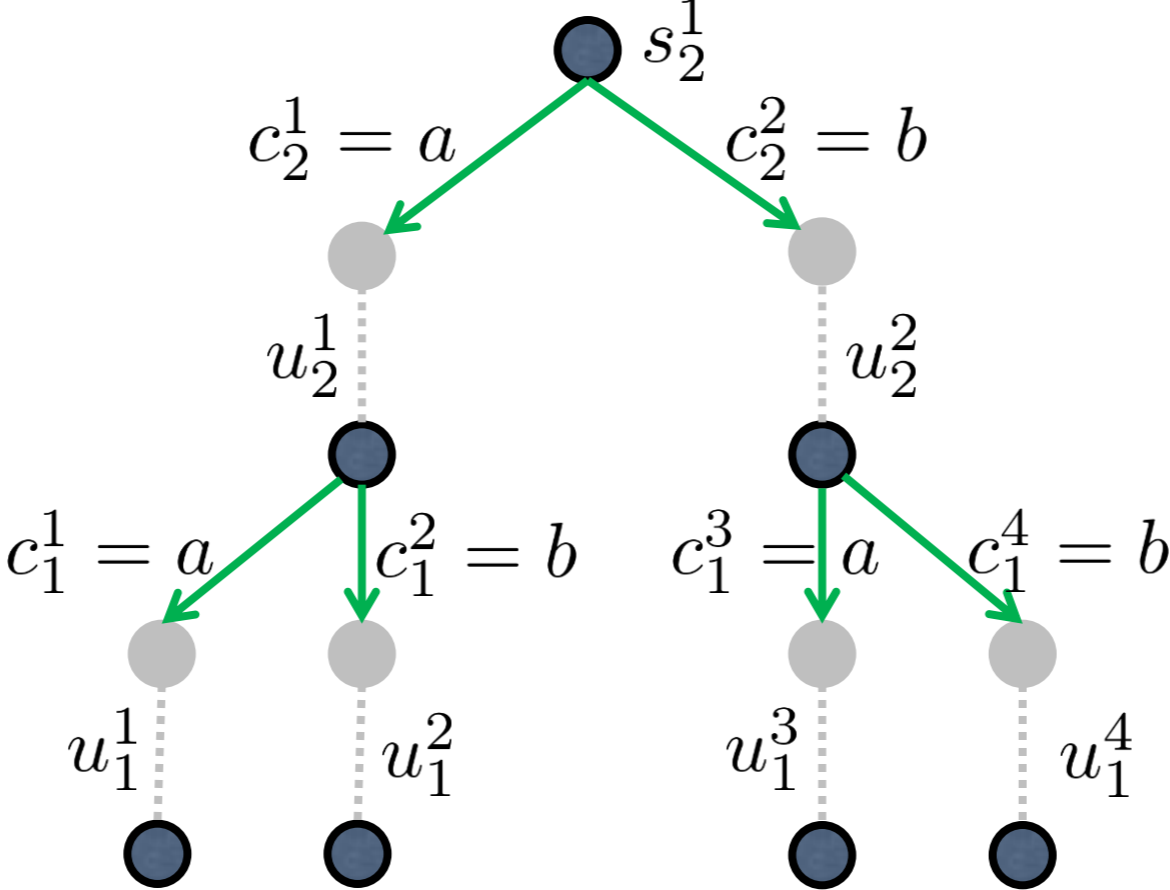
Abstract Game tree verification



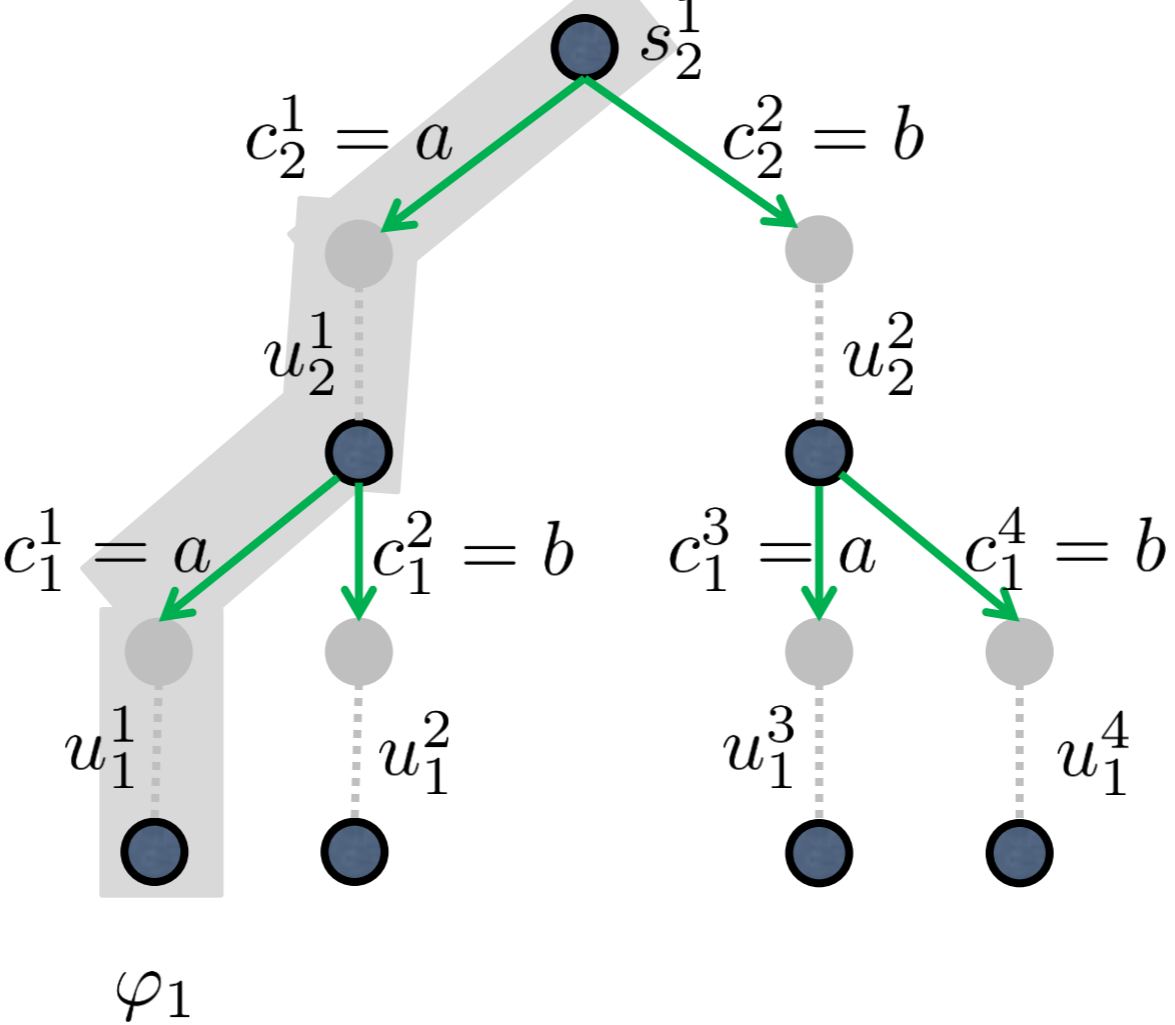
Abstract Game tree verification



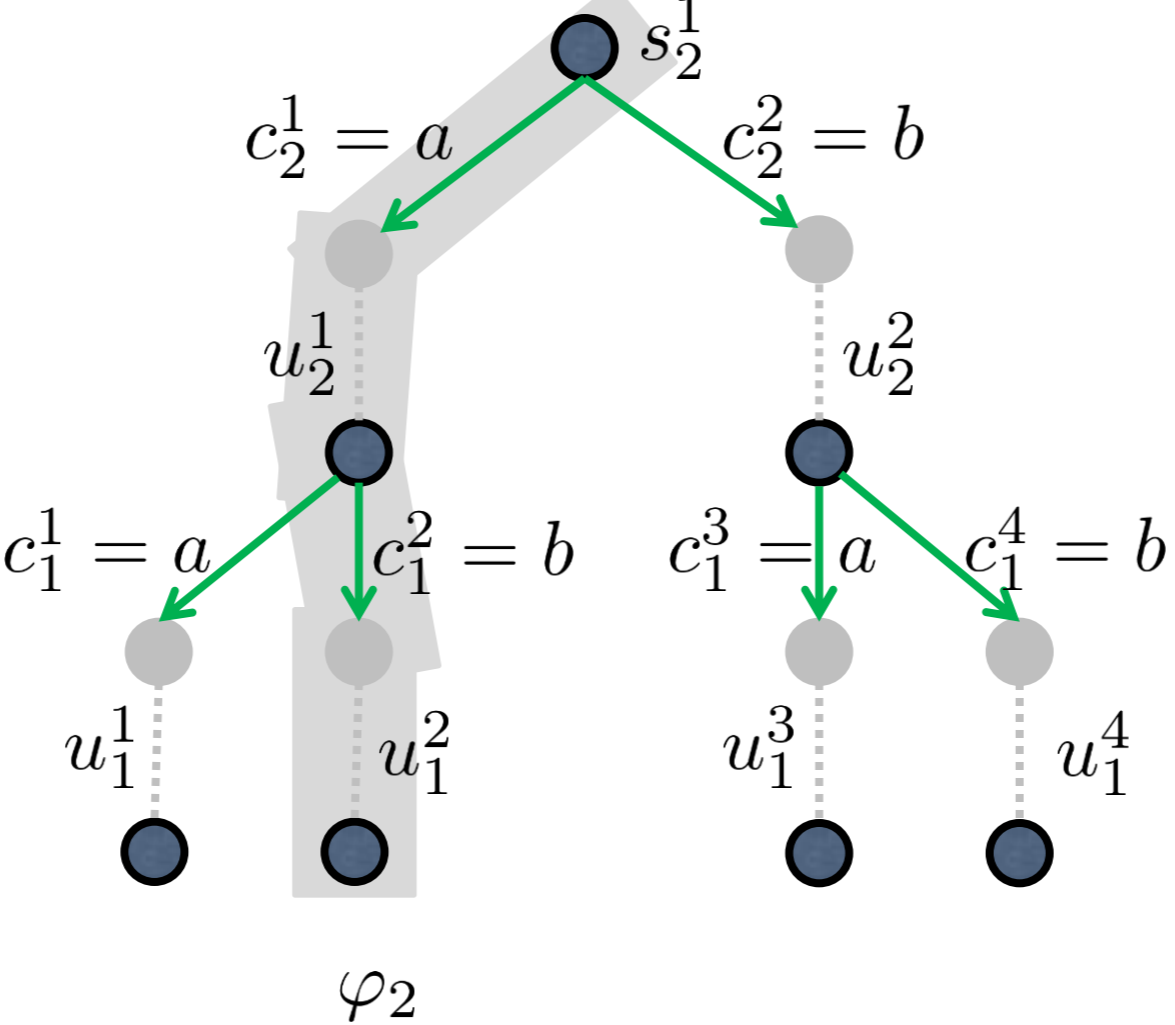
Abstract Game tree verification



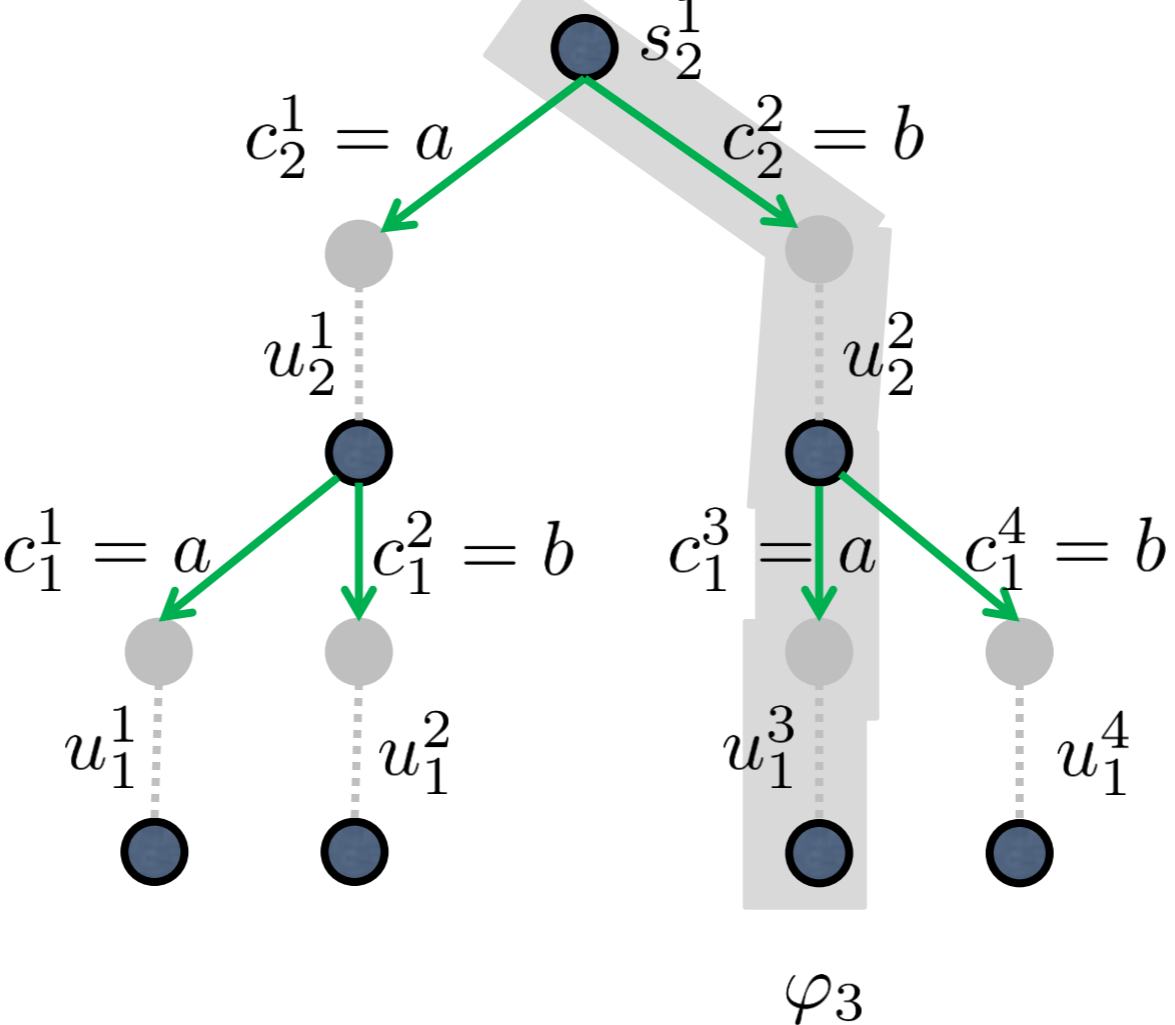
Abstract Game tree verification



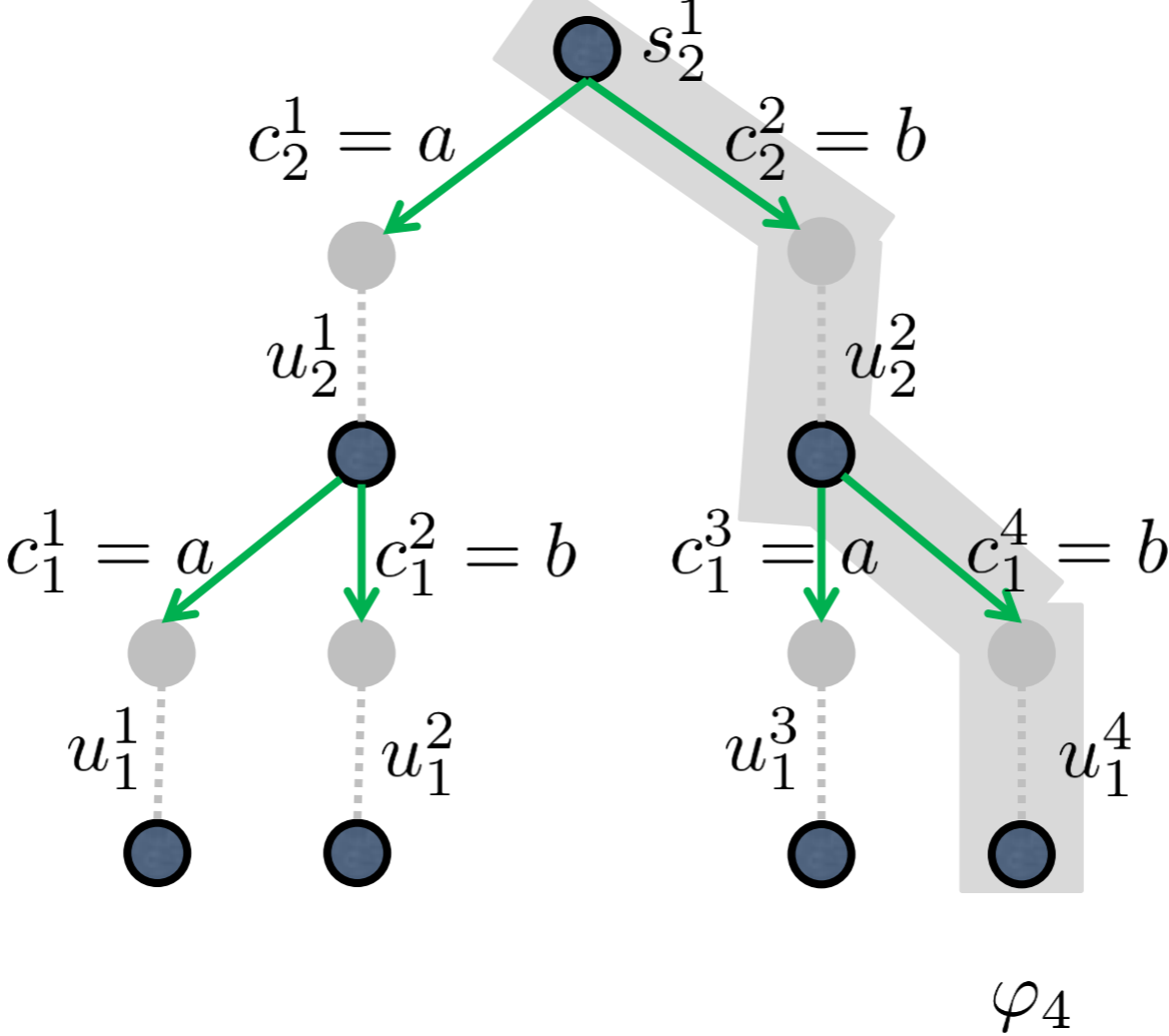
Abstract Game tree verification



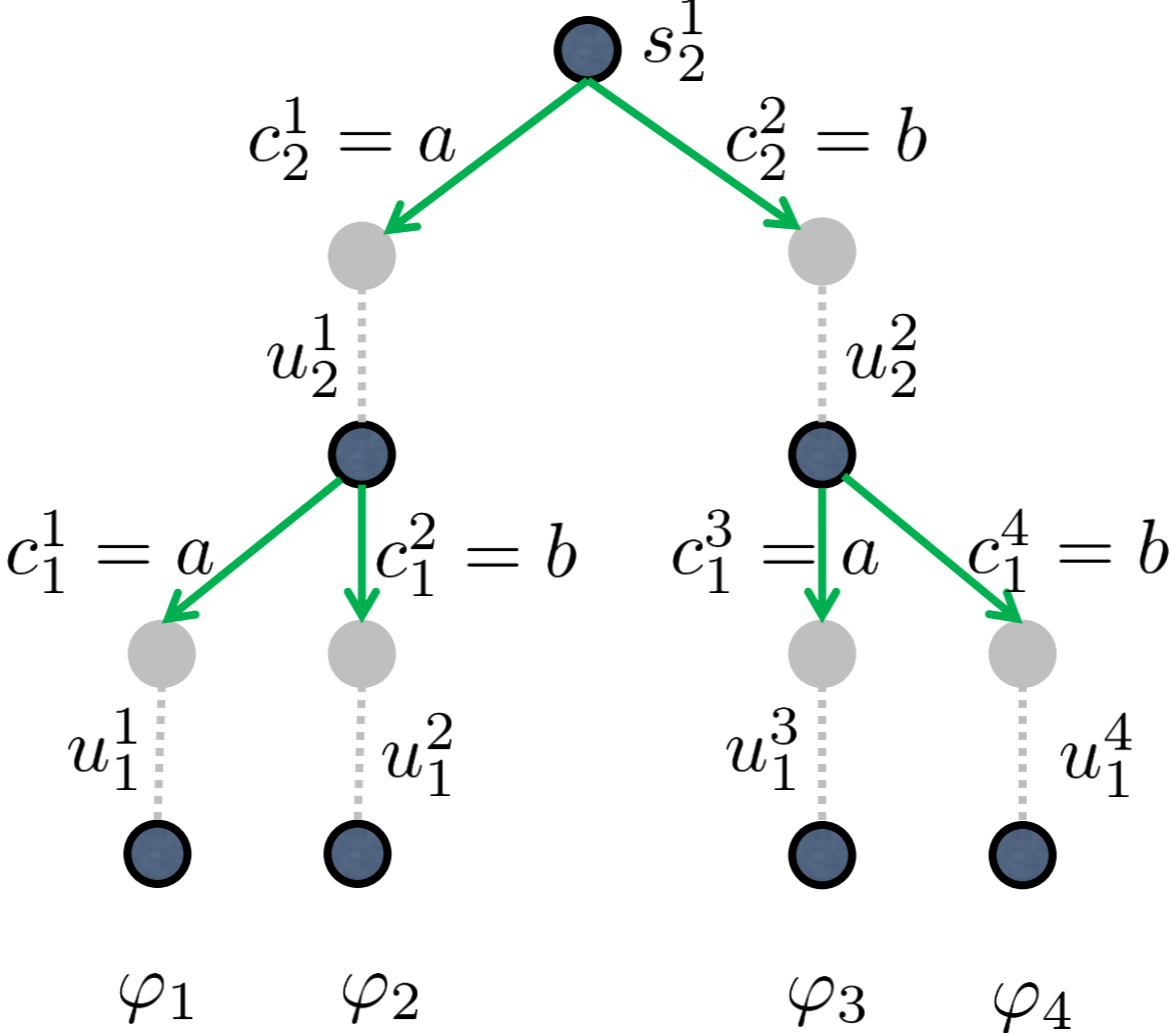
Abstract Game tree verification



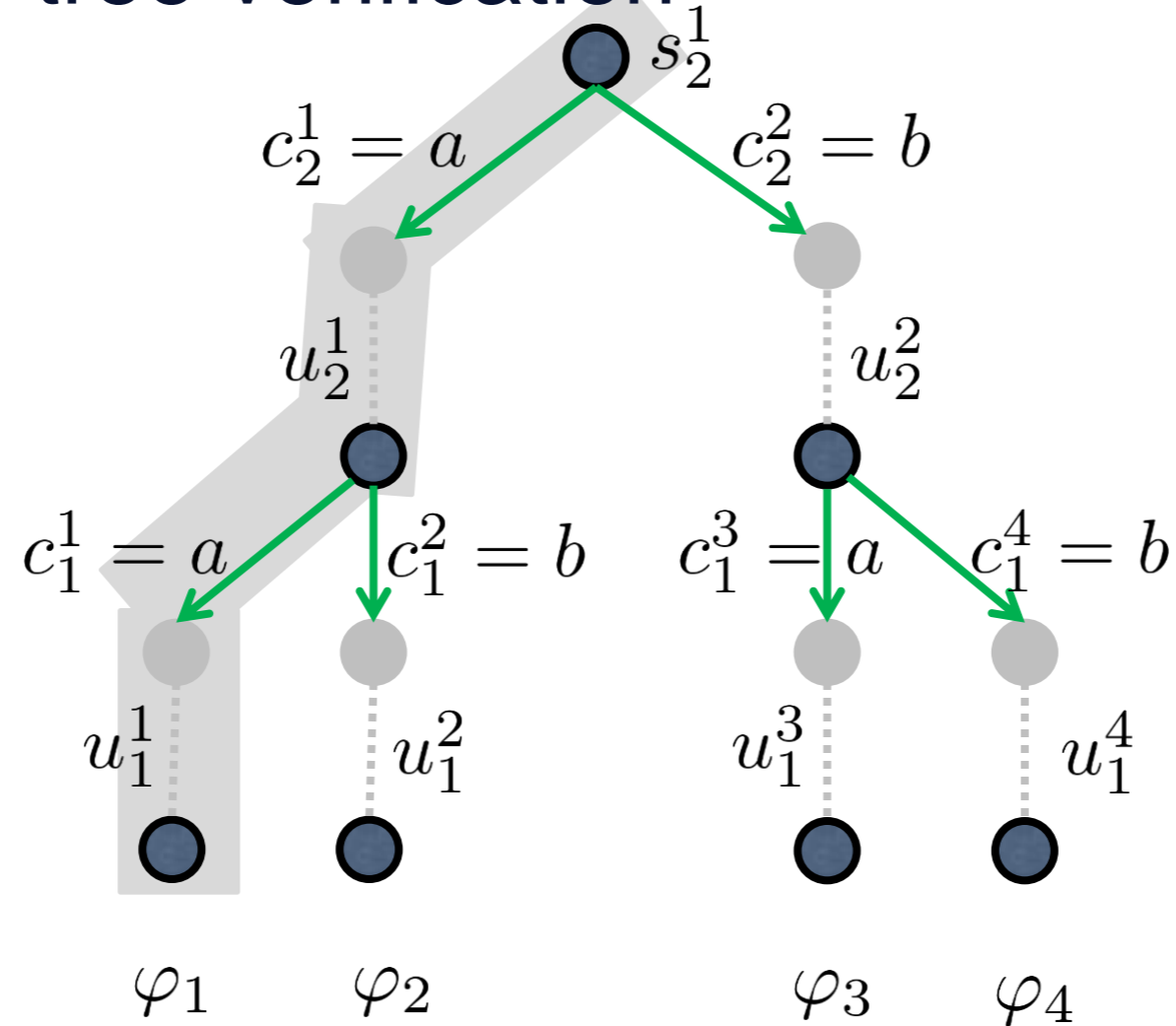
Abstract Game tree verification



Abstract Game tree verification



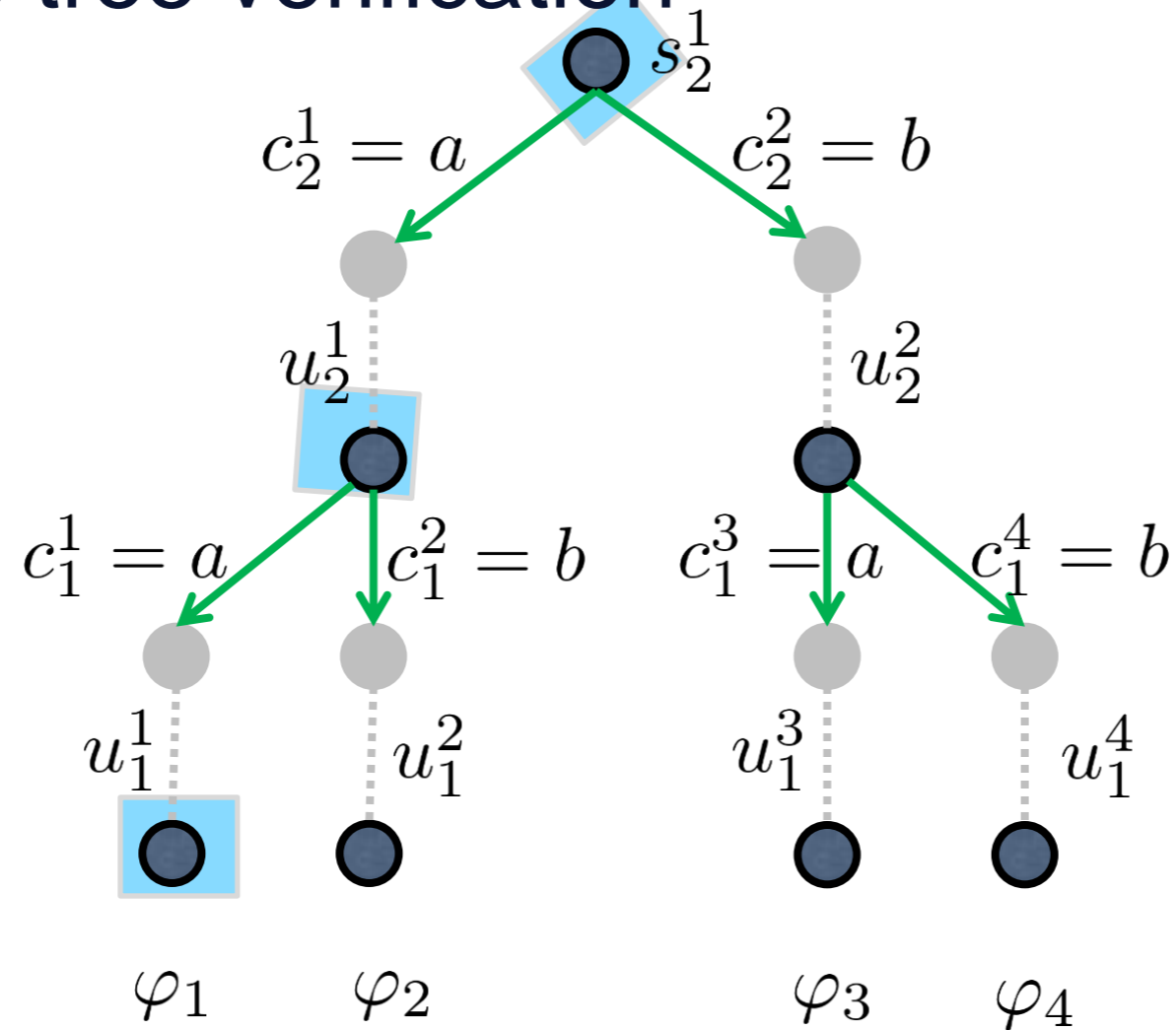
Abstract Game tree verification



$$\varphi_1 = \neg G(s_2^1) \wedge \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge \neg G(s_1^1) \wedge \delta(s_1^1, c_1^1, u_1^1, s_0^1) \wedge (c_1^1 = a) \wedge \neg G(s_0^1)$$



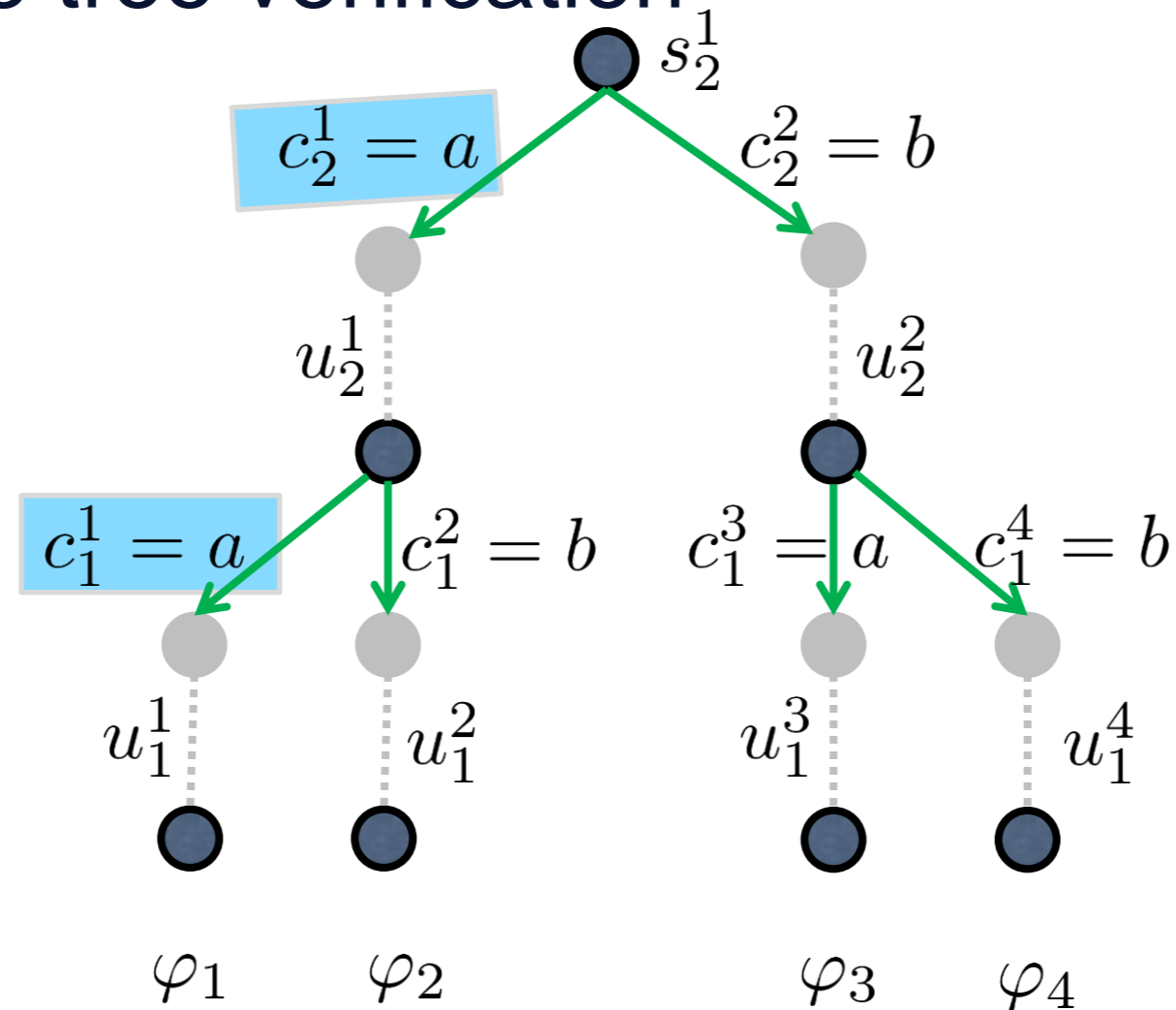
Abstract Game tree verification



$$\varphi_1 = \neg G(s_2^1) \wedge \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge \neg G(s_1^1) \wedge \delta(s_1^1, c_1^1, u_1^1, s_0^1) \wedge (c_1^1 = a) \wedge \neg G(s_0^1)$$



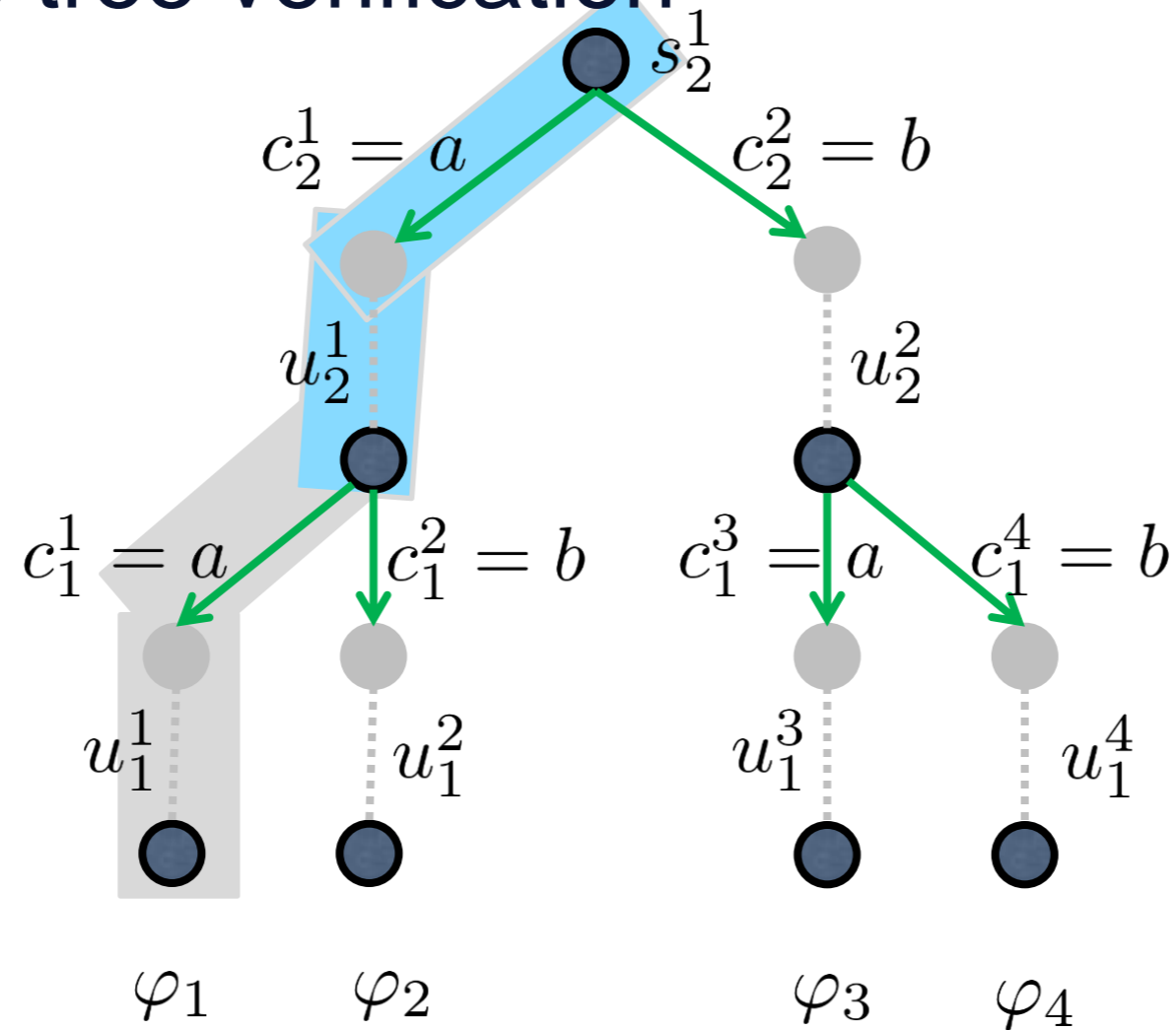
Abstract Game tree verification



$$\varphi_1 = \neg G(s_2^1) \wedge \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge \neg G(s_1^1) \wedge \delta(s_1^1, c_1^1, u_1^1, s_0^1) \wedge (c_1^1 = a) \wedge \neg G(s_0^1)$$



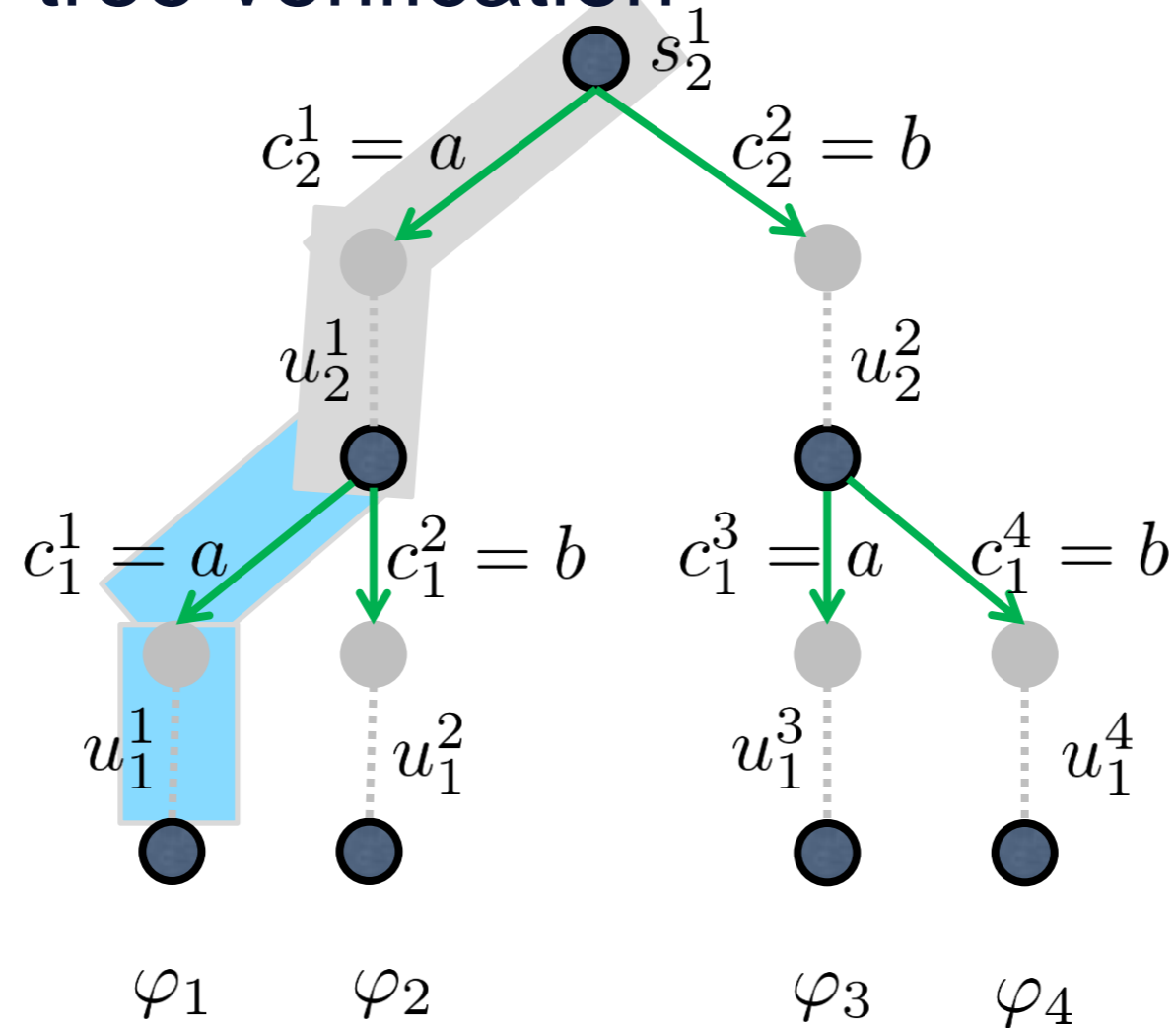
Abstract Game tree verification



$$\varphi_1 = \neg G(s_2^1) \wedge \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge \neg G(s_1^1) \wedge \delta(s_1^1, c_1^1, u_1^1, s_0^1) \wedge (c_1^1 = a) \wedge \neg G(s_0^1)$$



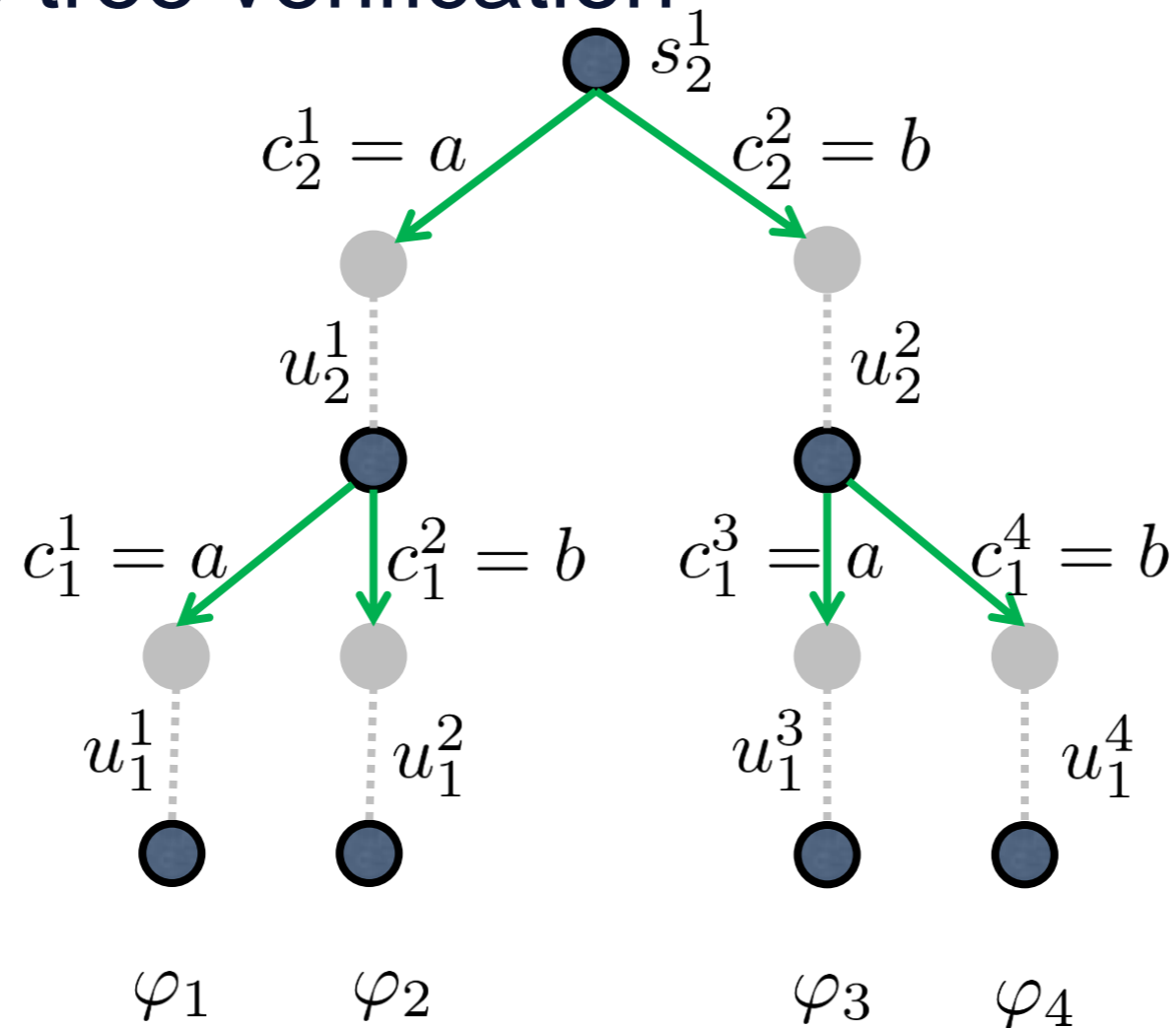
Abstract Game tree verification



$$\varphi_1 = \neg G(s_2^1) \wedge \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge \neg G(s_1^1) \wedge \delta(s_1^1, c_1^1, u_1^1, s_0^1) \wedge (c_1^1 = a) \wedge \neg G(s_0^1)$$



Abstract Game tree verification



$$\varphi_1 = \neg G(s_2^1) \wedge \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge \neg G(s_1^1) \wedge \delta(s_1^1, c_1^1, u_1^1, s_0^1) \wedge (c_1^1 = a) \wedge \neg G(s_0^1)$$

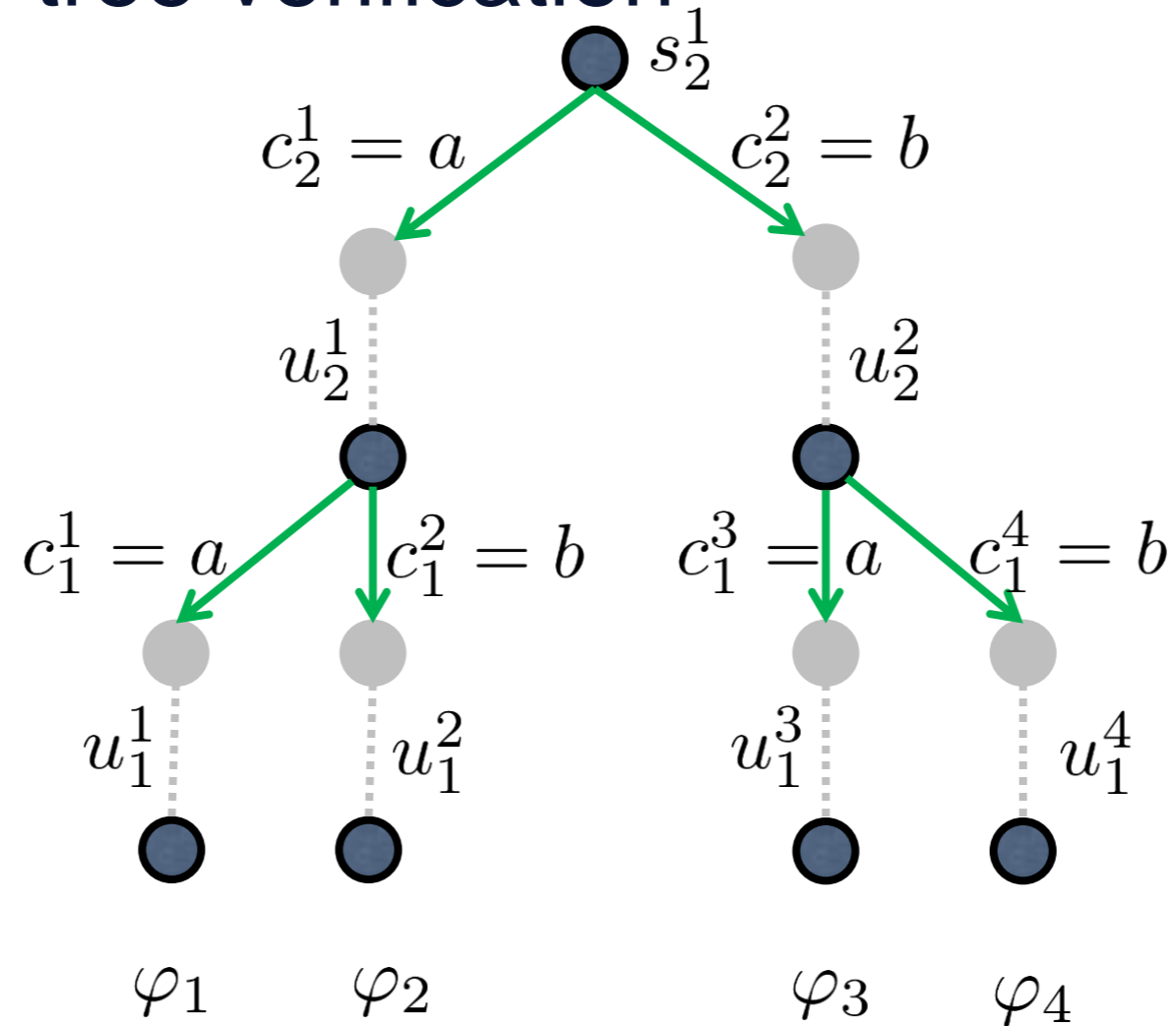
$$\varphi_2 = \dots$$

$$\varphi_3 = \dots$$

$$\varphi_4 = \dots$$



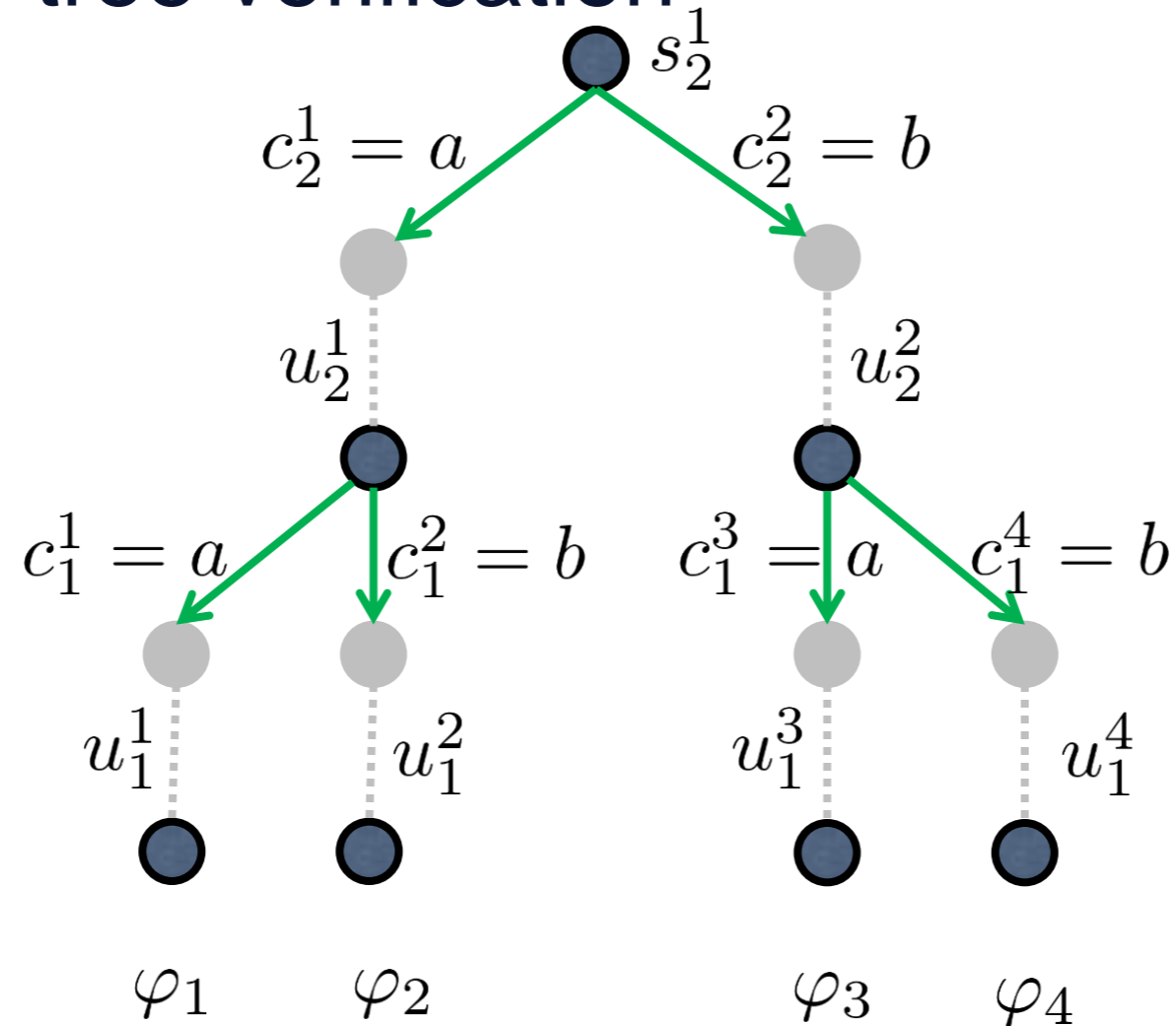
Abstract Game tree verification



$$(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$$



Abstract Game tree verification



$$(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$$

is

UNSAT

(by correctness of EvaSolver)



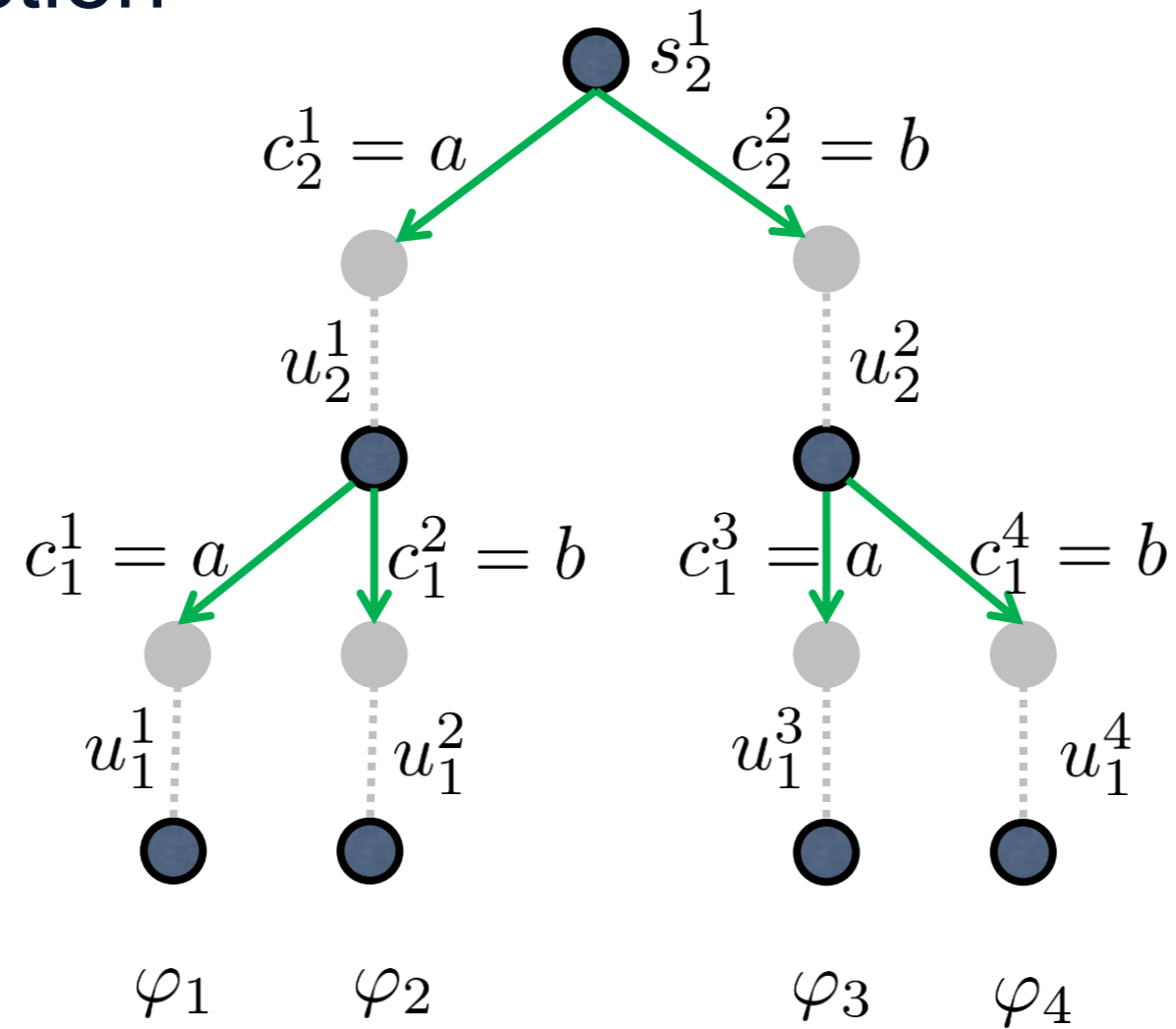
How to extract a winning strategy



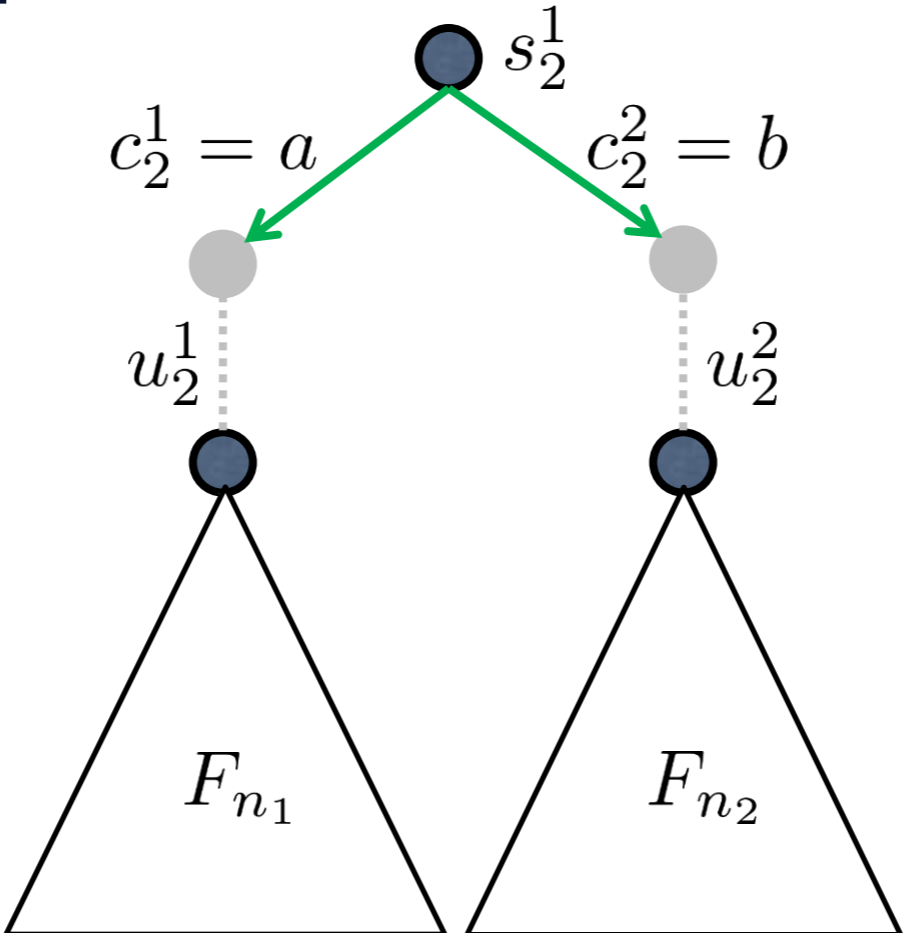
Partitioning operation



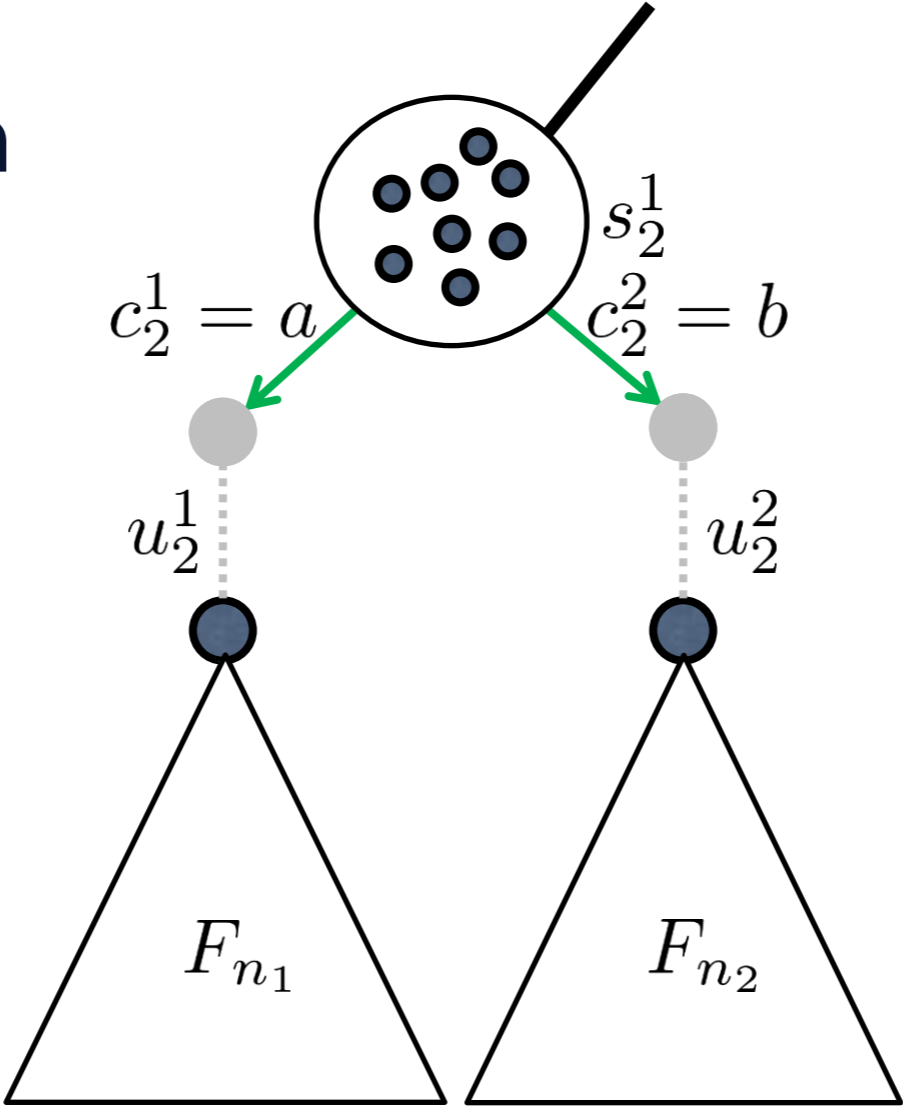
Strategy extraction



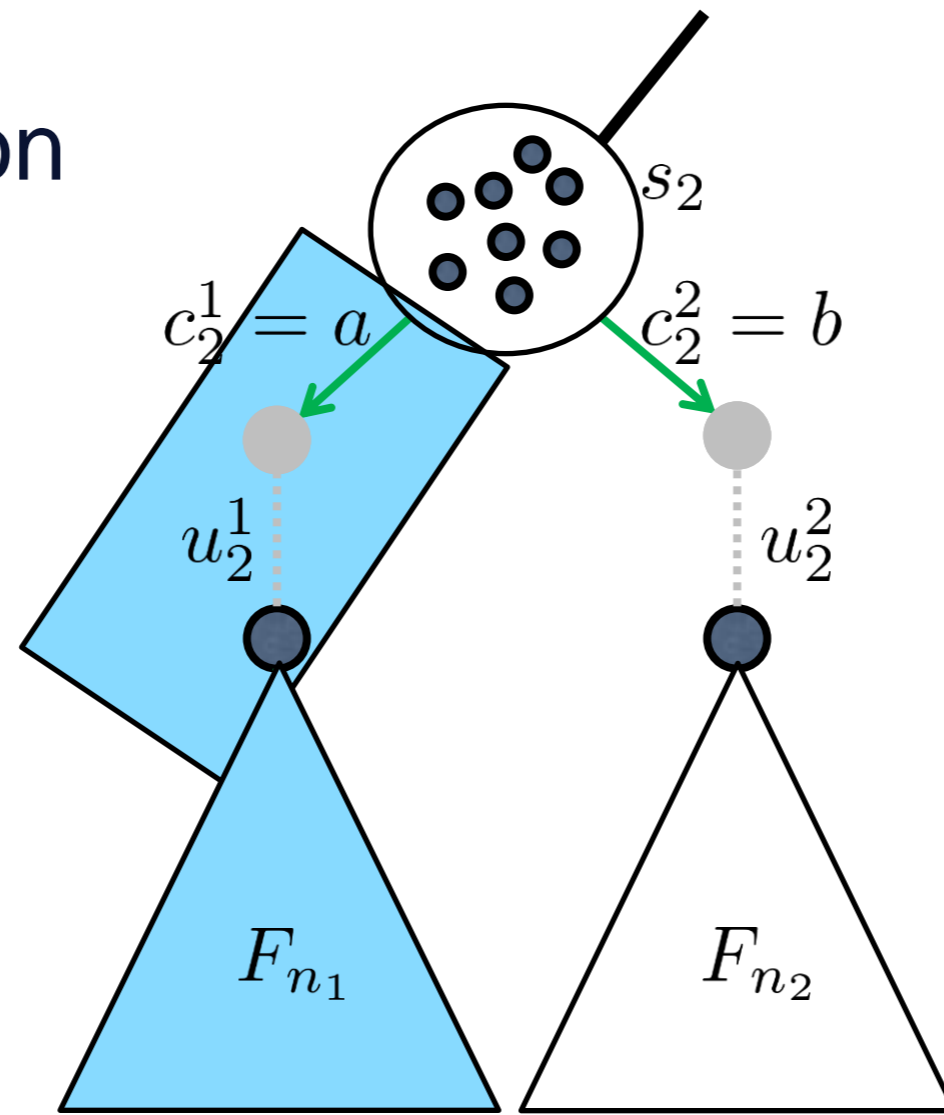
Strategy extraction



Strategy extraction



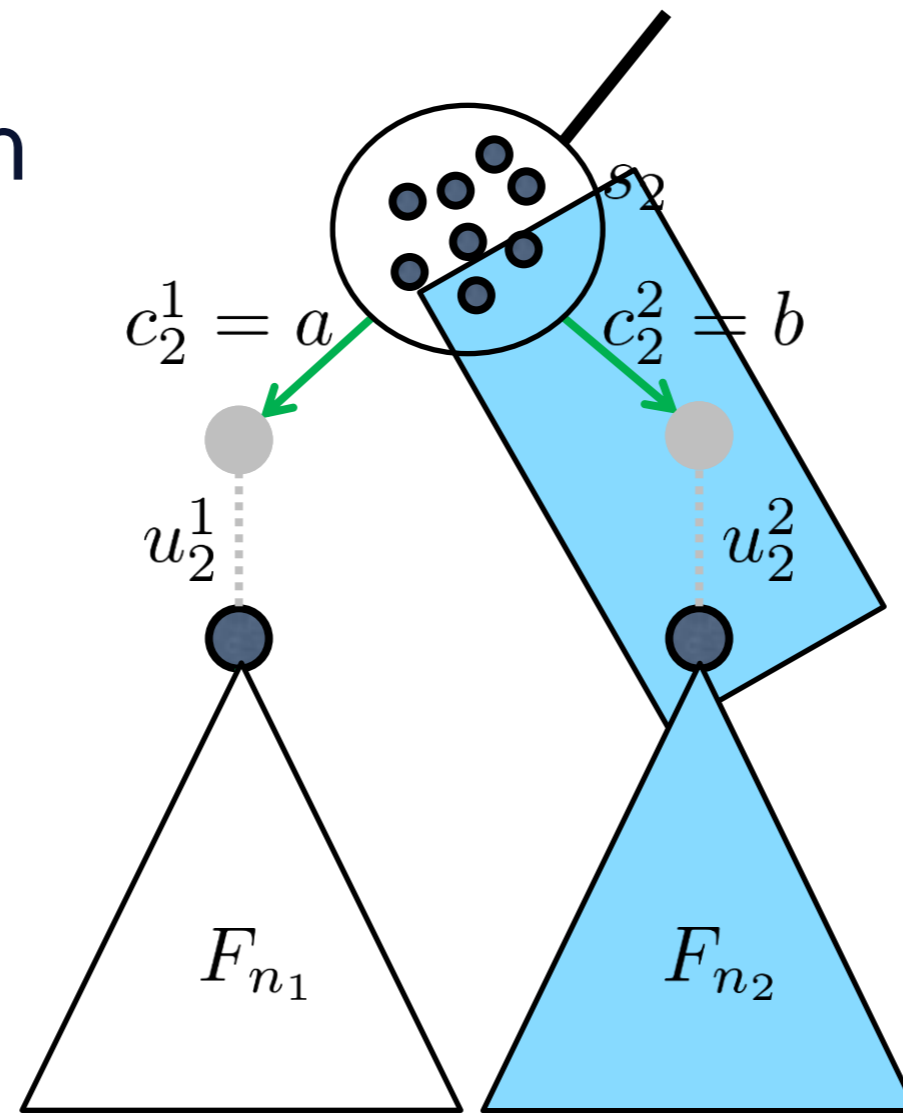
Strategy extraction



$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$



Strategy extraction

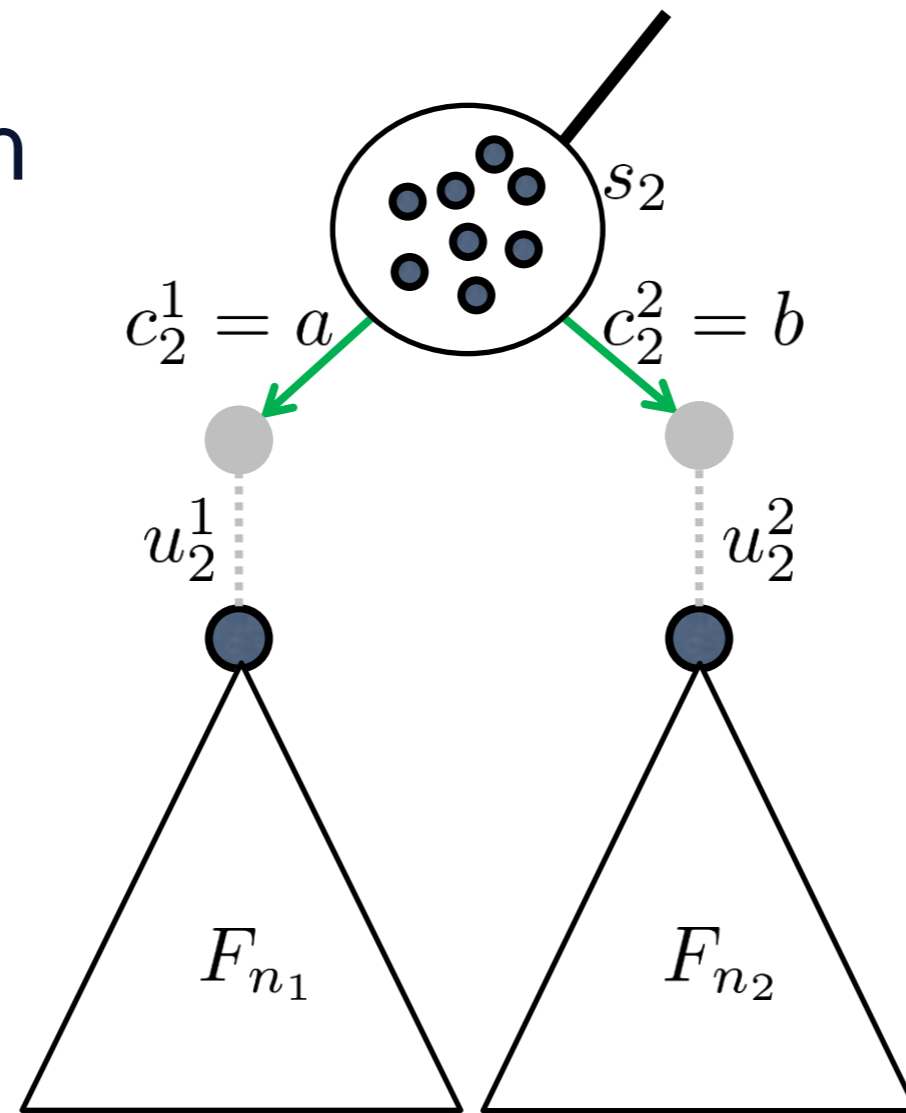


$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^1) \wedge (c_2^2 = b) \wedge F_{n_2}.$$



Strategy extraction



$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b]$$



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^2, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$



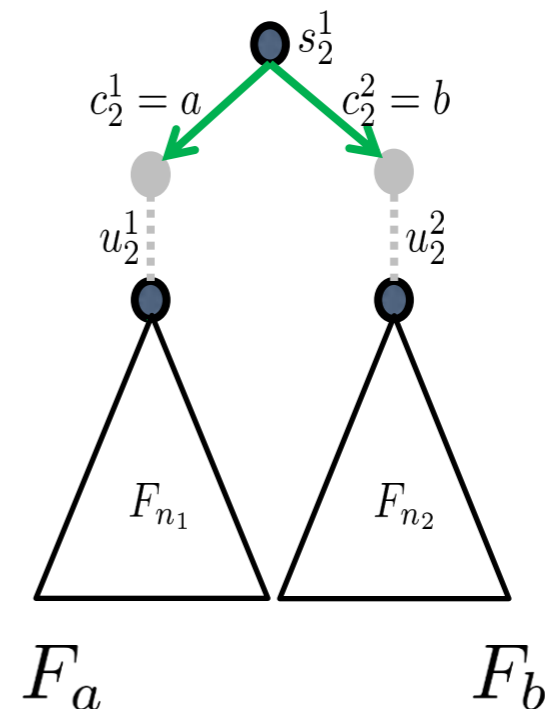
Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^1) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$



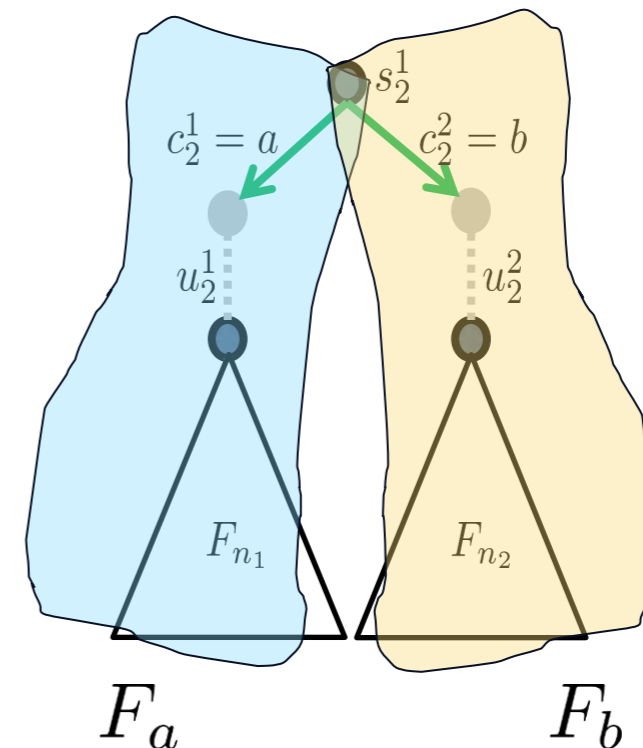
Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

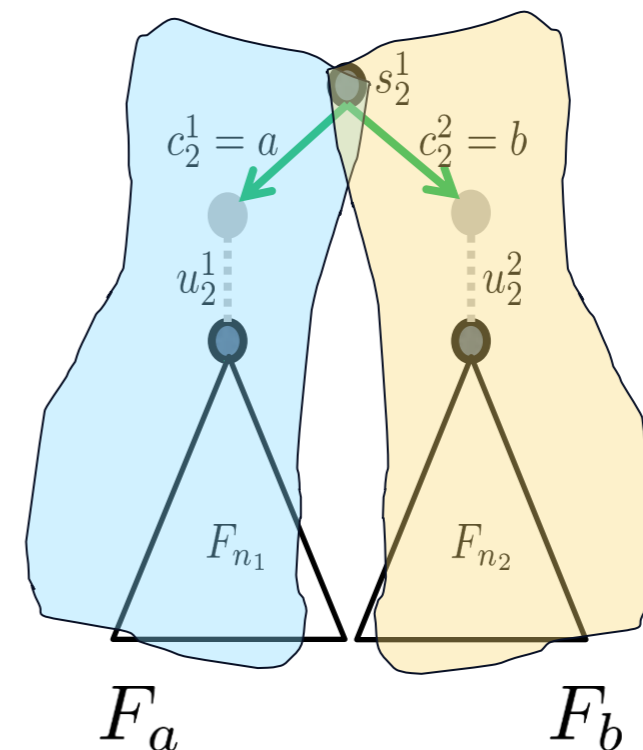
$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^1) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$

$$\mathcal{I}(s_2^1) = \text{Interpolant}(F_a, F_b):$$

- $F_a \implies \mathcal{I}$
- $\mathcal{I} \wedge F_b = \perp$



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

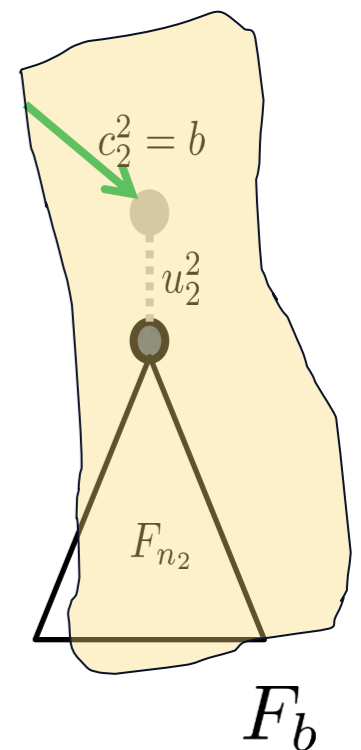
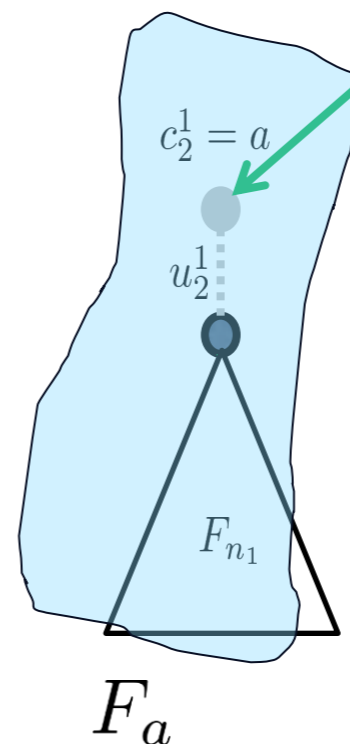
$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$

$$\mathcal{I}(s_2^1) = \text{Interpolant}(F_a, F_b):$$

- $F_a \implies \mathcal{I}$
- $\mathcal{I} \wedge F_b = \perp$

s_2^1
●



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

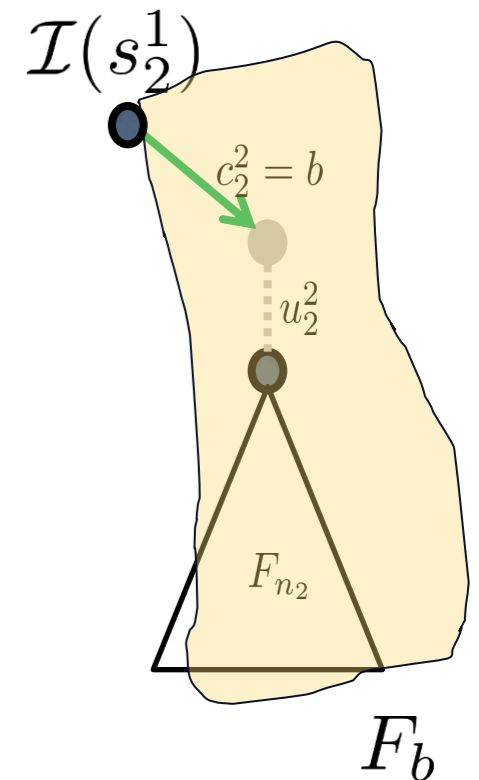
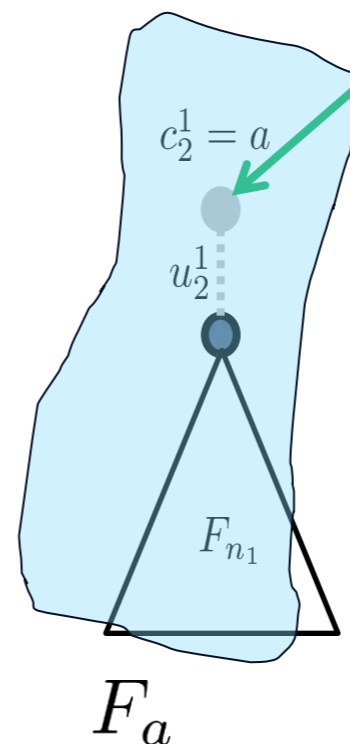
$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$

$$\mathcal{I}(s_2^1) = \text{Interpolant}(F_a, F_b):$$

- $F_a \implies \mathcal{I}$
- $\mathcal{I} \wedge F_b = \perp$

s_2^1



Strategy extraction

$$F_a = \delta(s_2^1, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a) \wedge F_{n_1}.$$

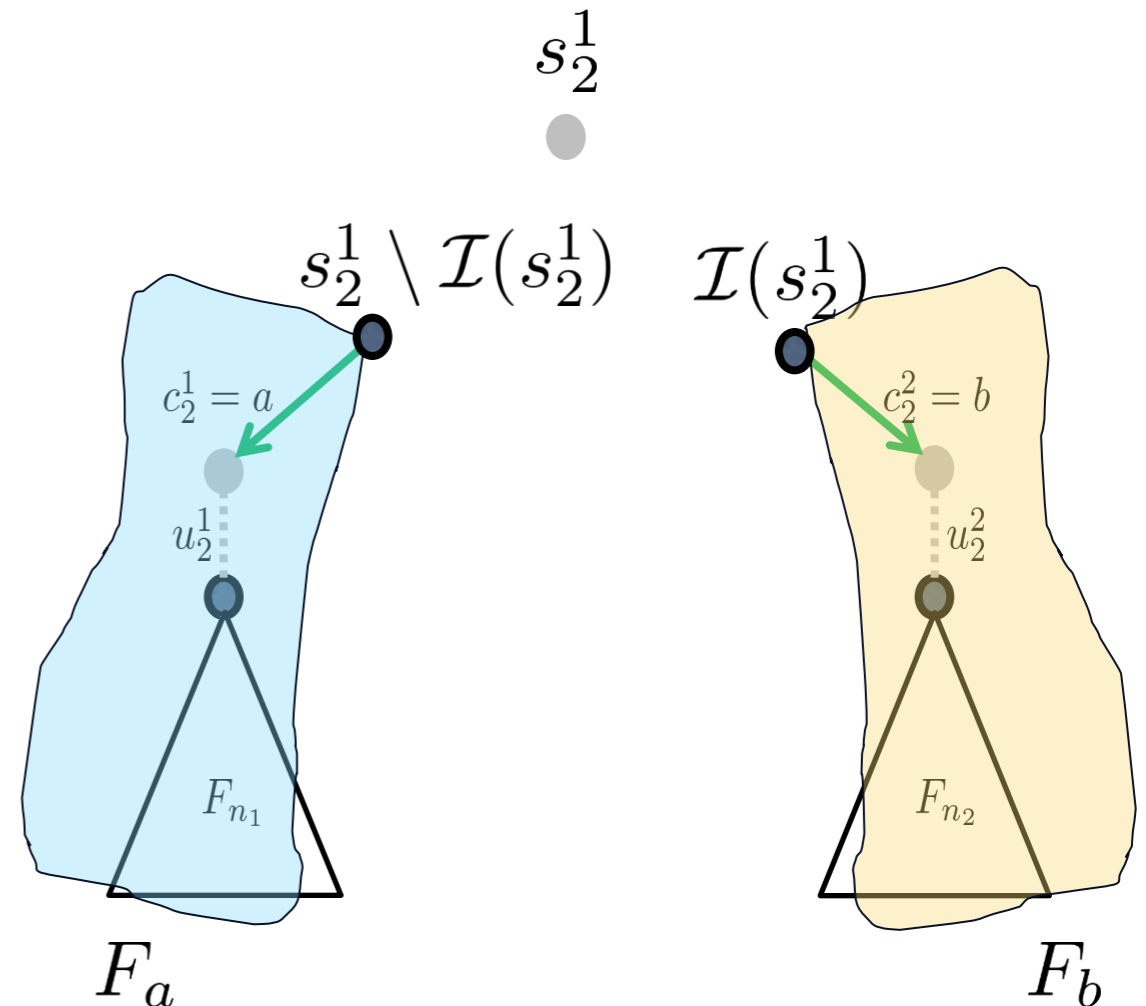
$$F_b = (s_2^1 = I) \wedge \neg G(s_2^1) \wedge \delta(s_2^1, c_2^2, u_2^2, s_1^2) \wedge (c_2^2 = b) \wedge F_{n_2}.$$

$$[(s_2^1 = I) \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4] \equiv [F_a \wedge F_b] = \perp$$

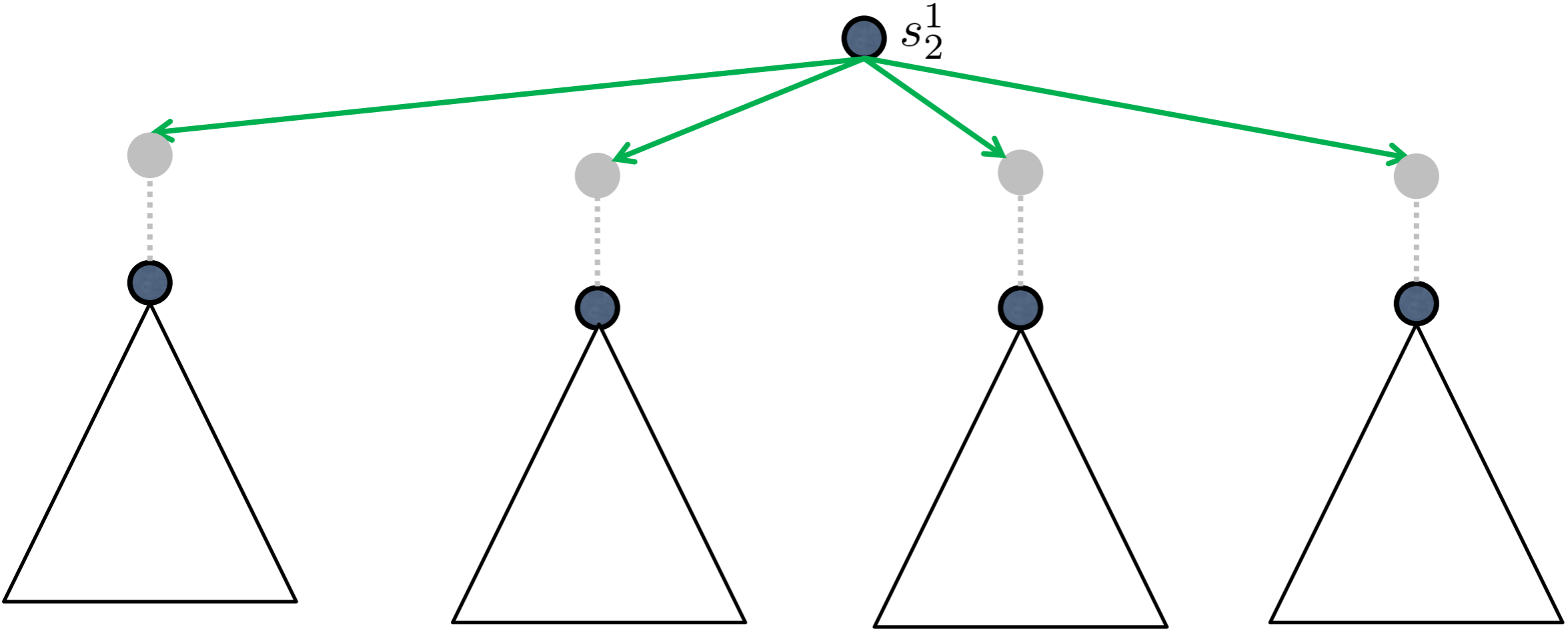
$$\text{vars}(F_a) \cap \text{vars}(F_b) = s_2^1$$

$$\mathcal{I}(s_2^1) = \text{Interpolant}(F_a, F_b):$$

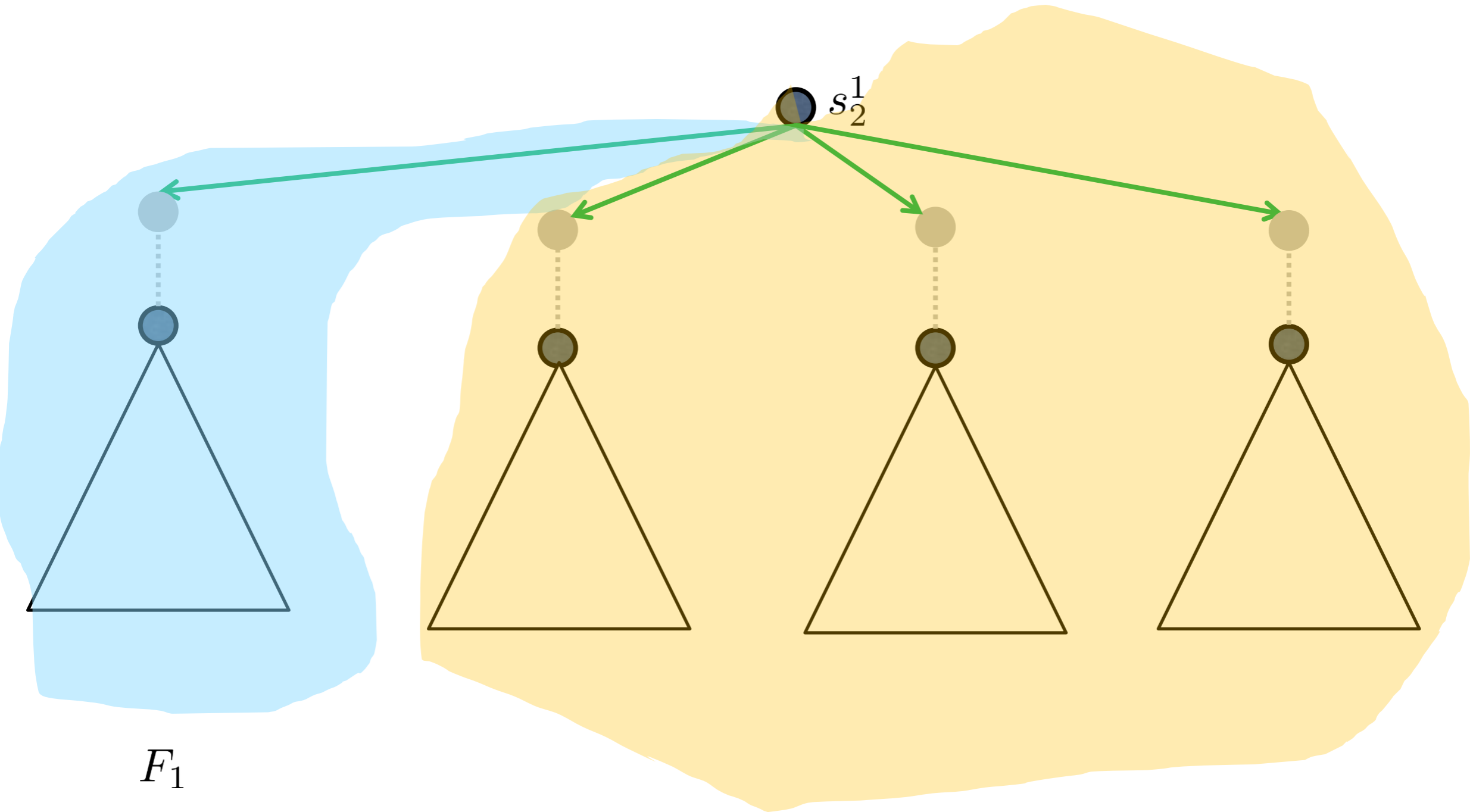
- $F_a \implies \mathcal{I}$
- $\mathcal{I} \wedge F_b = \perp$



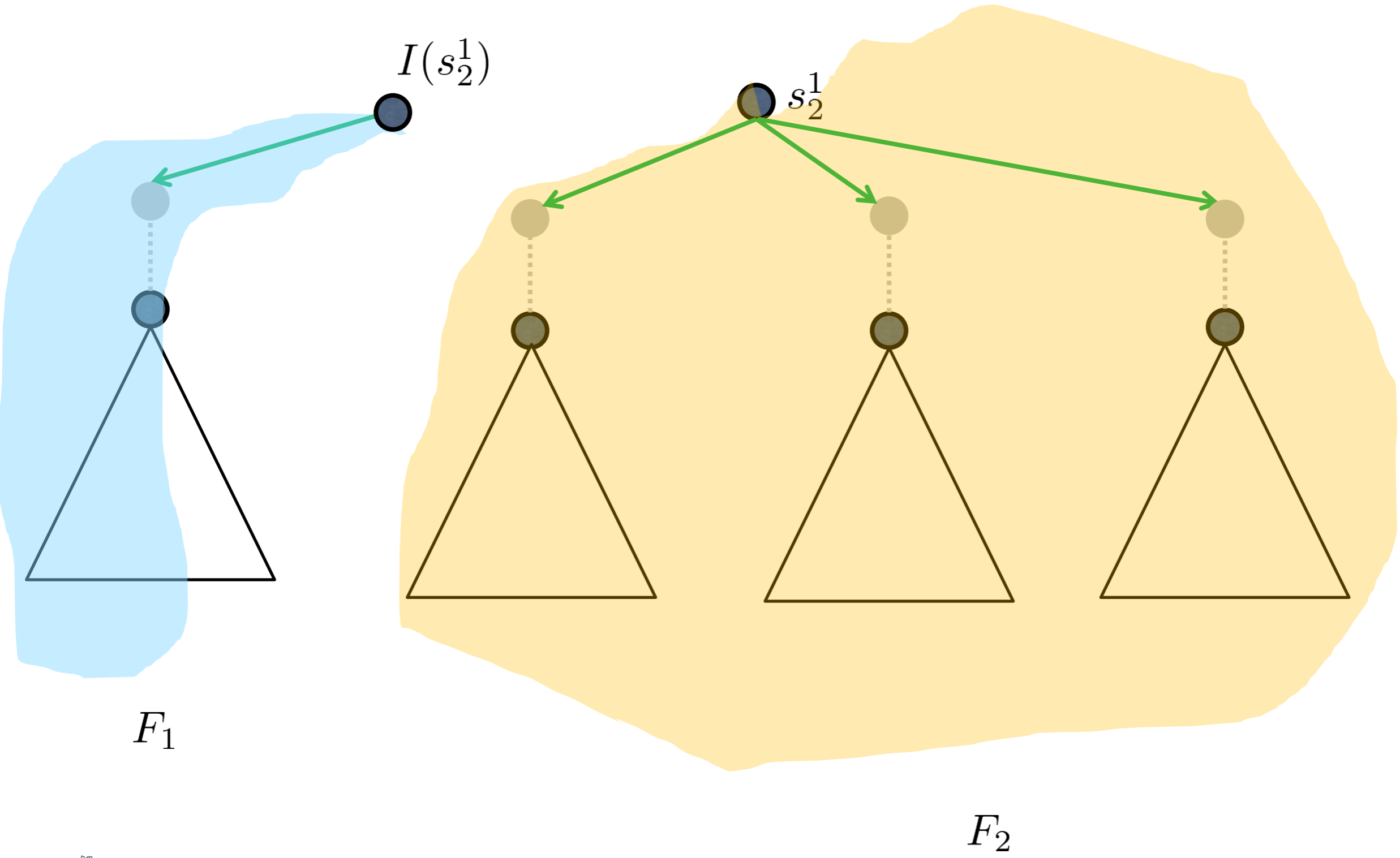
Strategy extraction



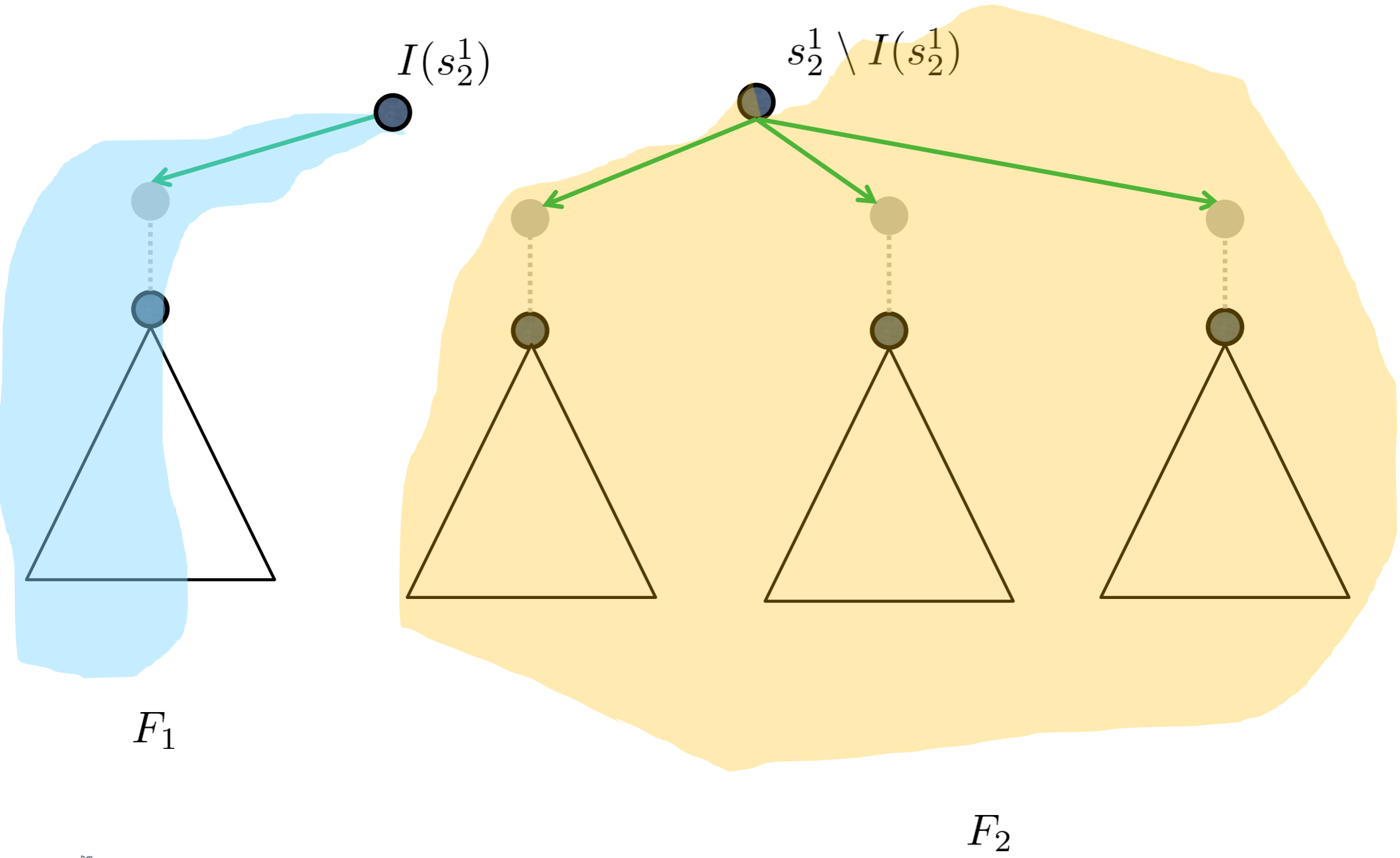
Strategy extraction



Strategy extraction



Strategy extraction



Size of interpolants

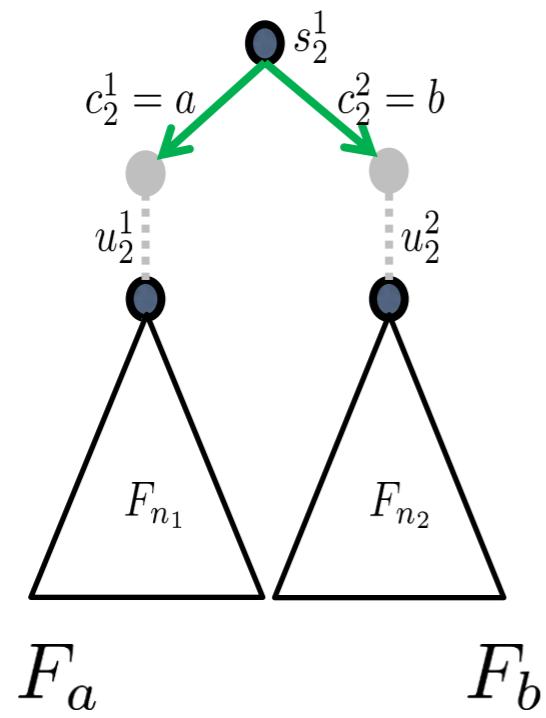
Size of interpolants is reasonable (tomorrow's talk).



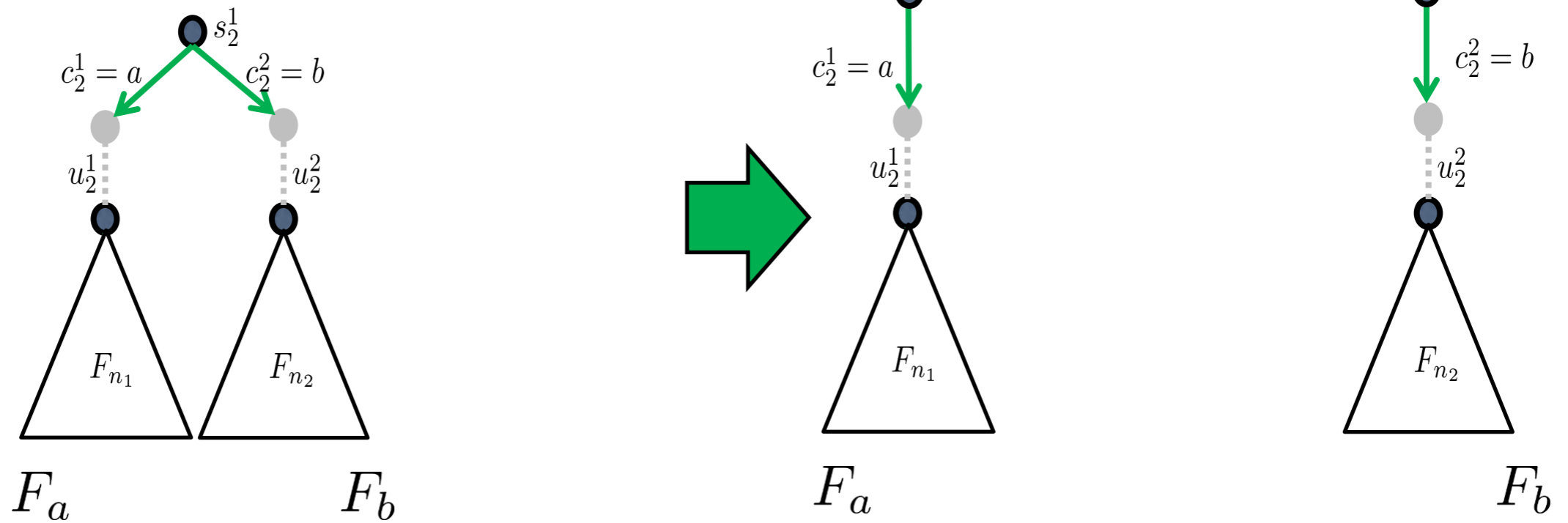
Next state operation



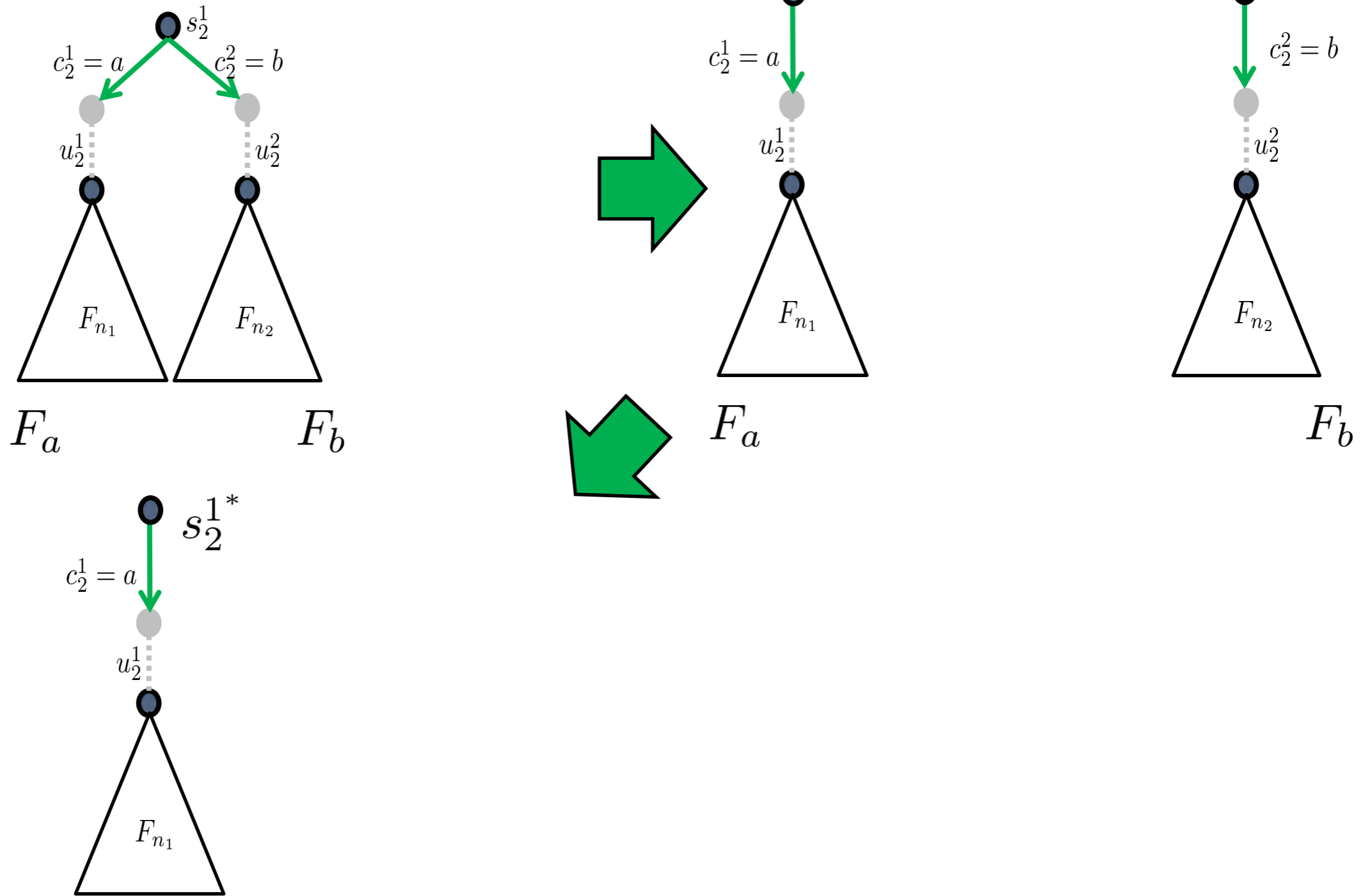
Strategy extraction



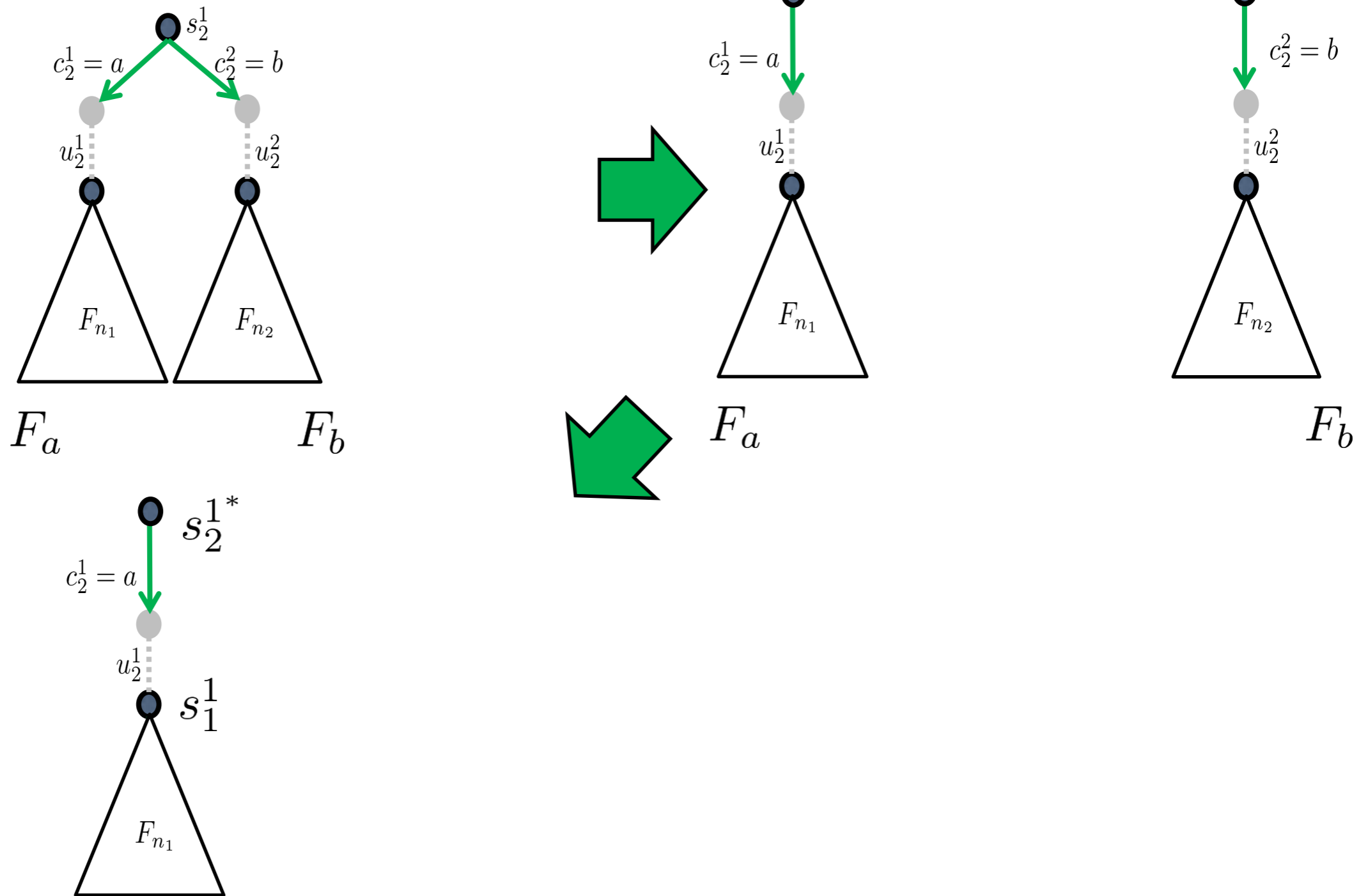
Strategy extraction



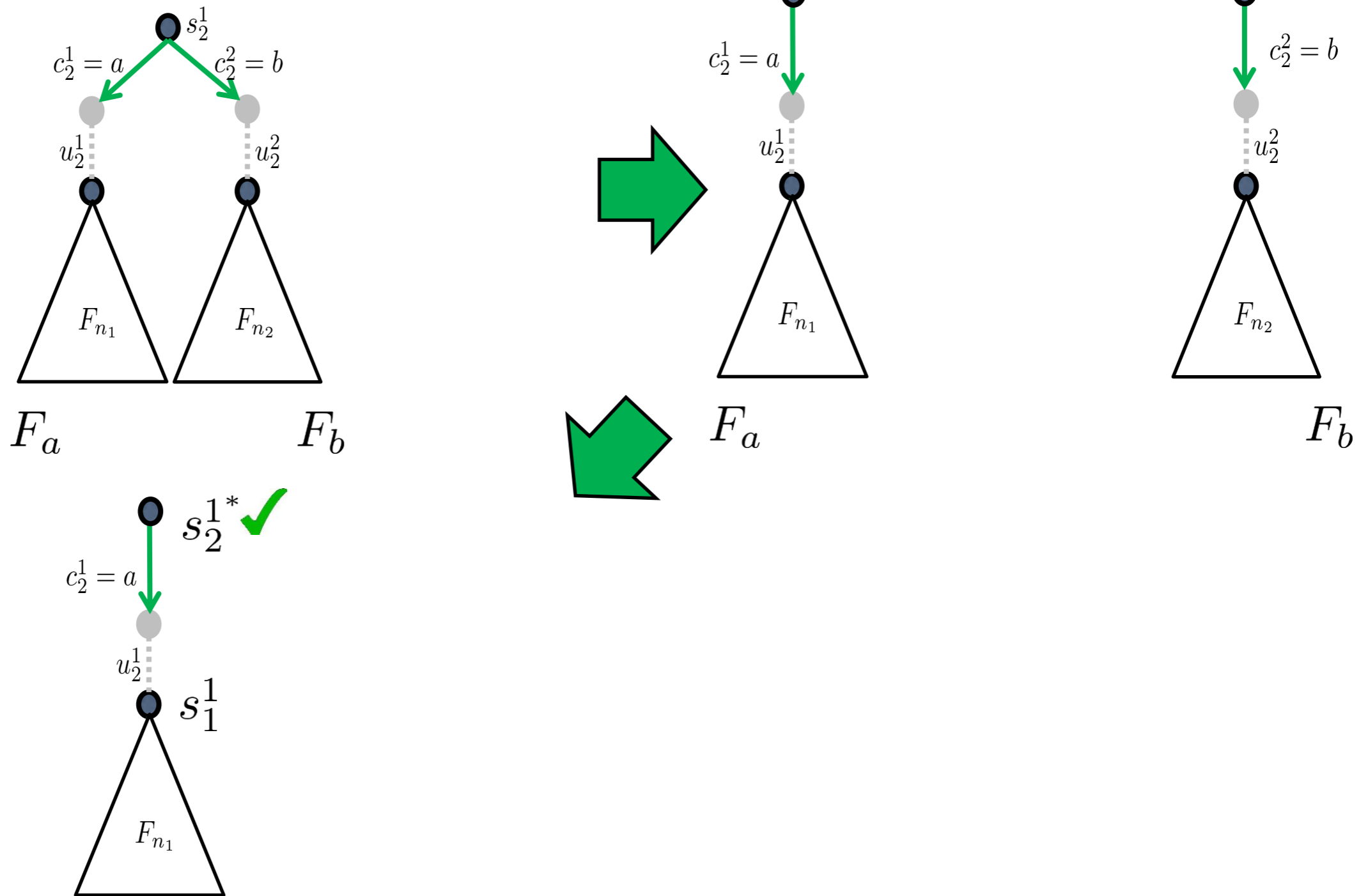
Strategy extraction



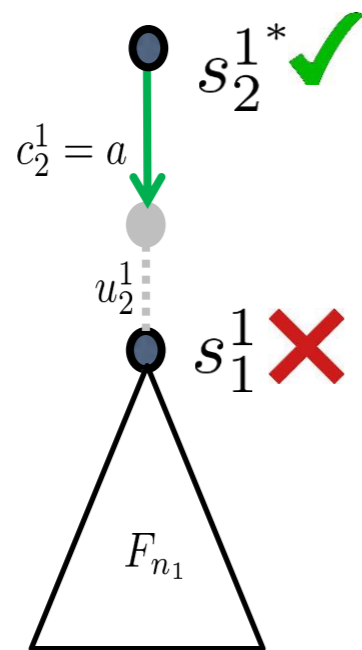
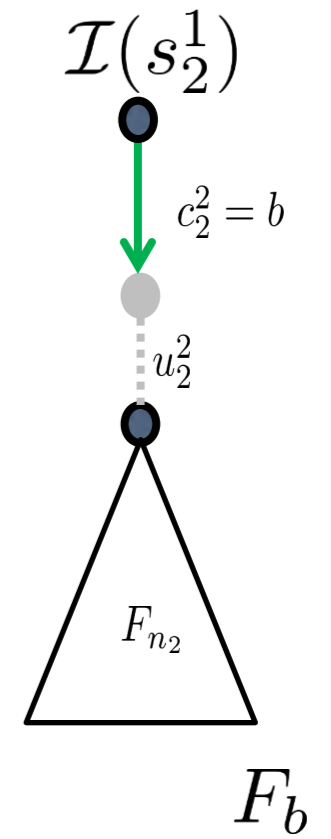
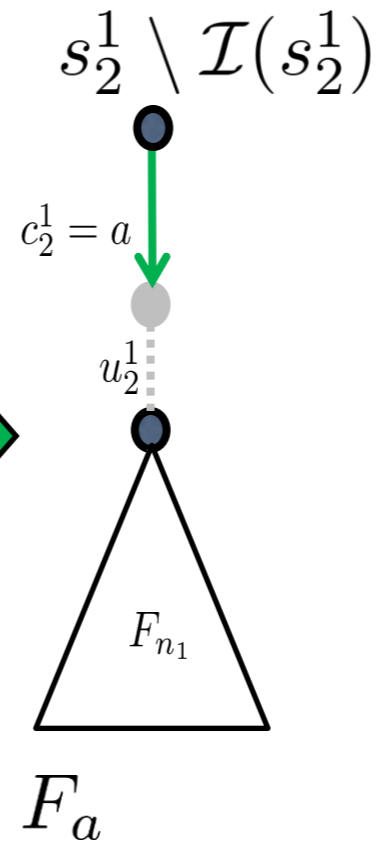
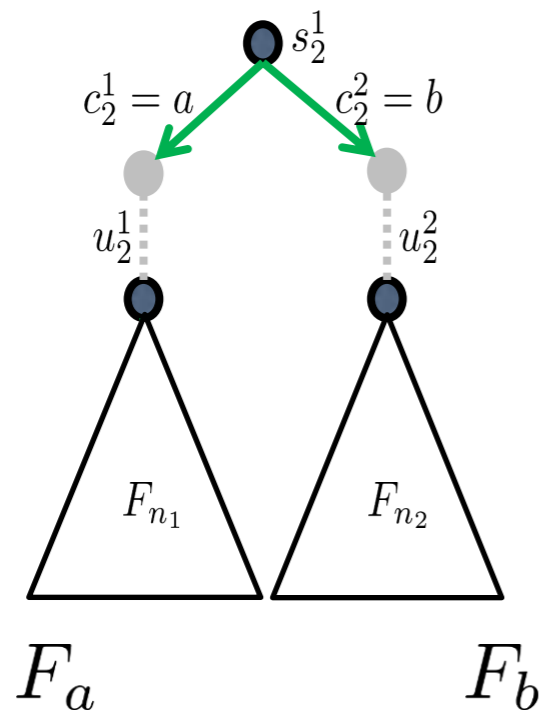
Strategy extraction



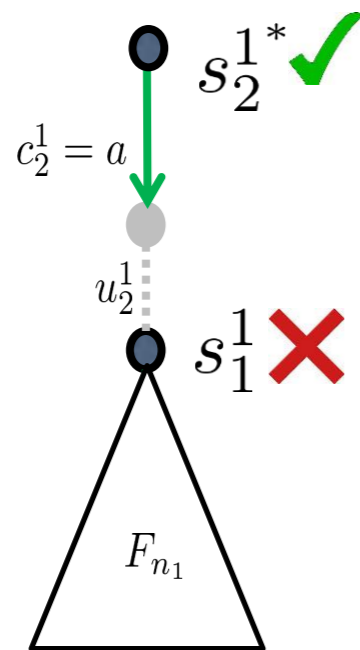
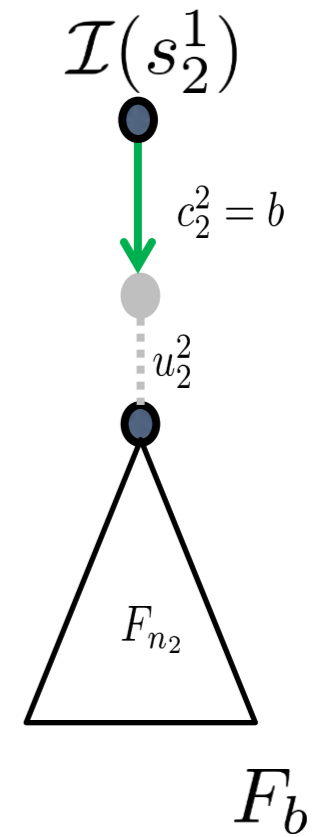
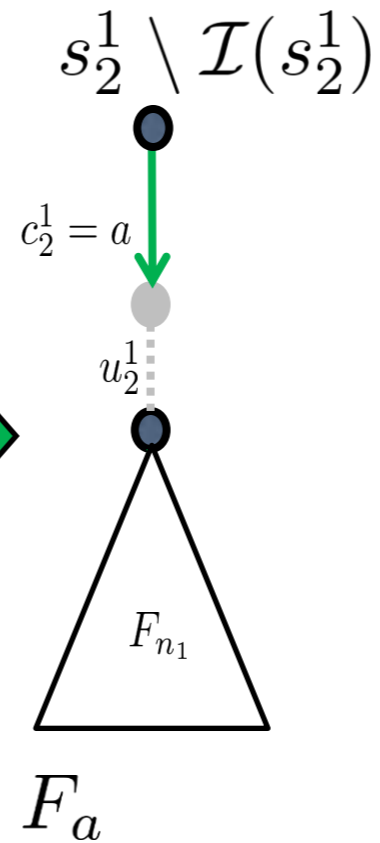
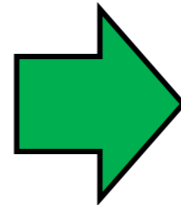
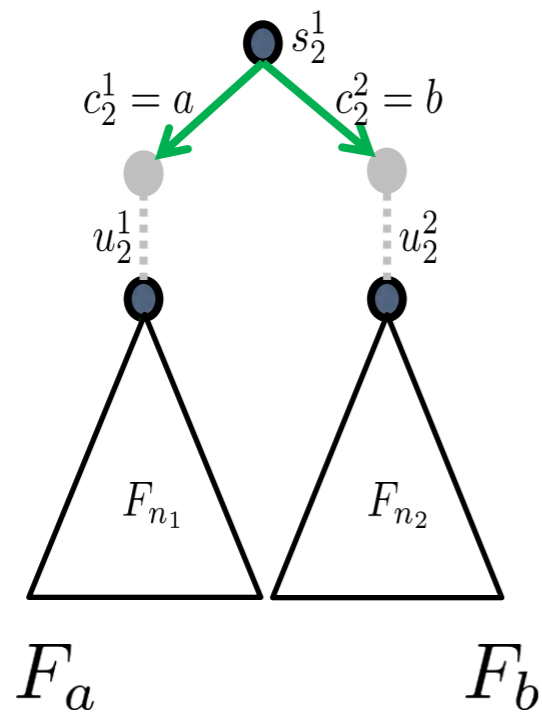
Strategy extraction



Strategy extraction



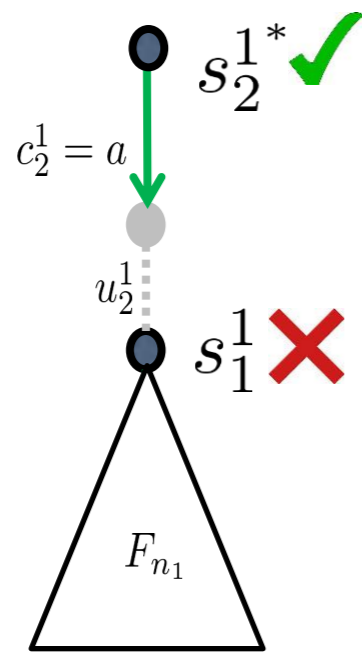
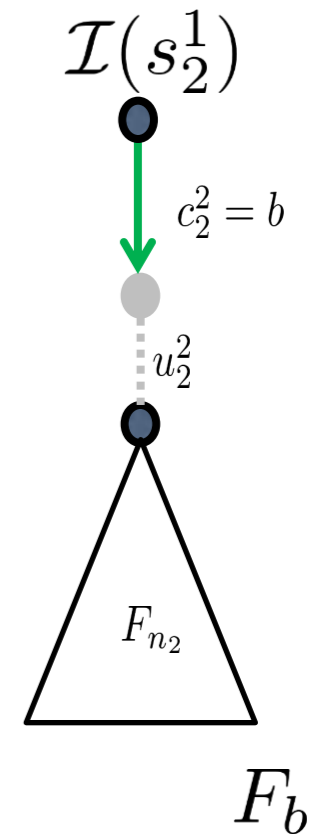
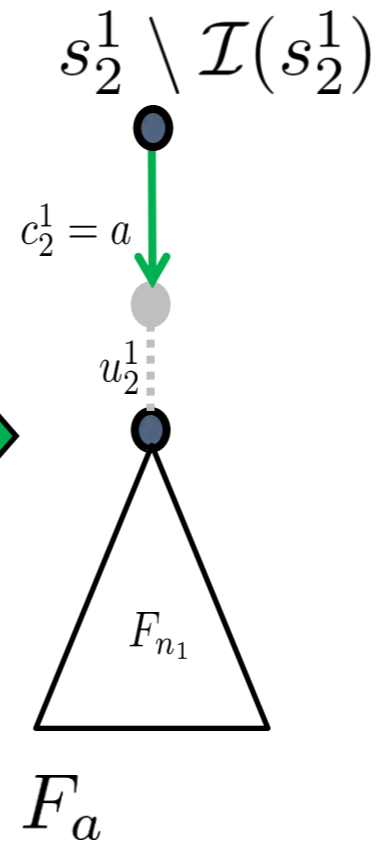
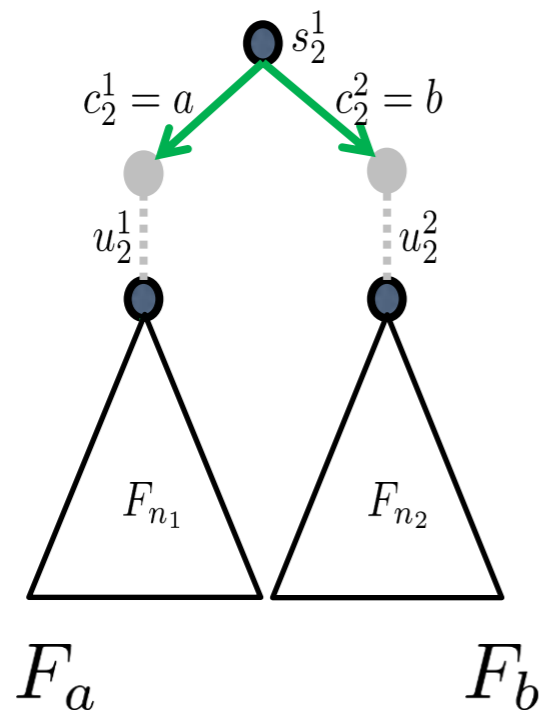
Strategy extraction



$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$



Strategy extraction

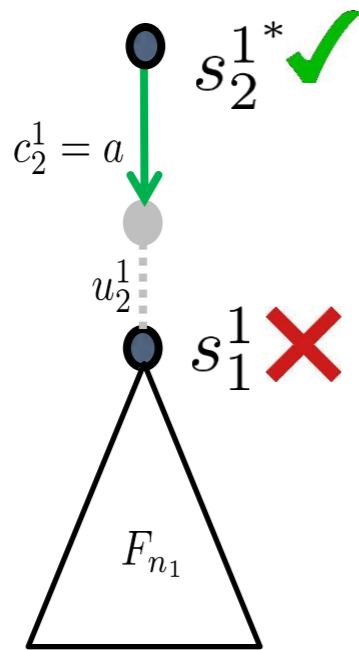
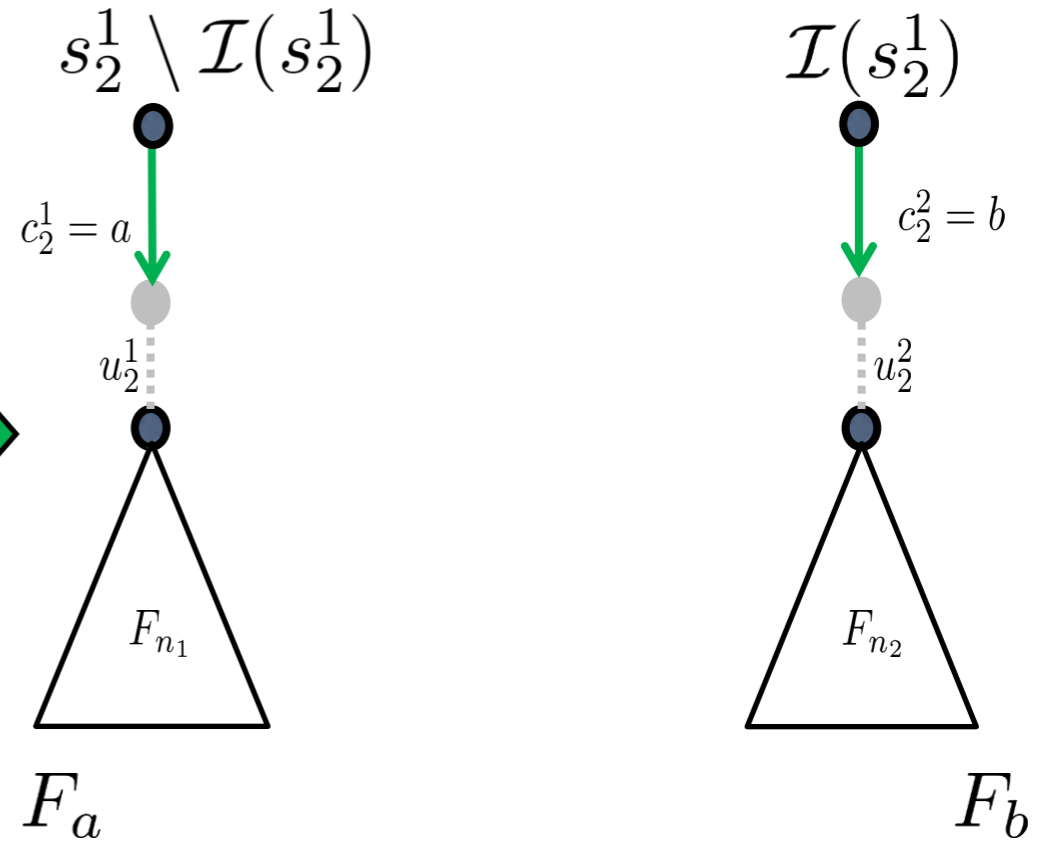
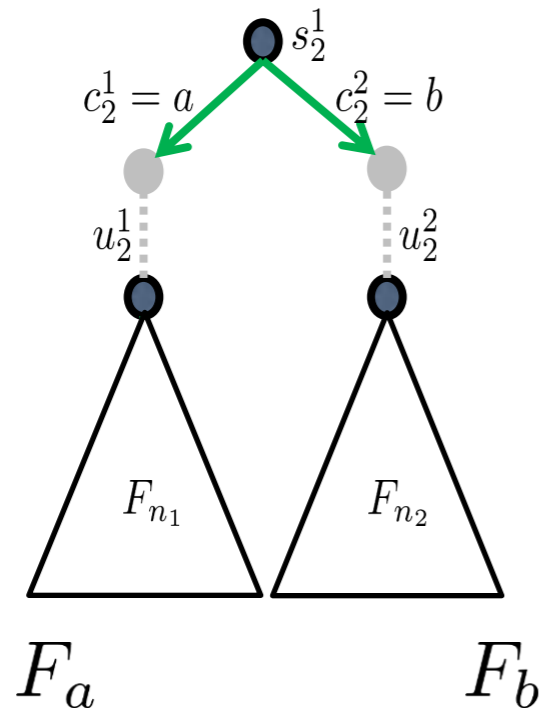


$$T = s_2^{1*} \wedge \delta(s_{n_1}, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

AllSAT(T)



Strategy extraction



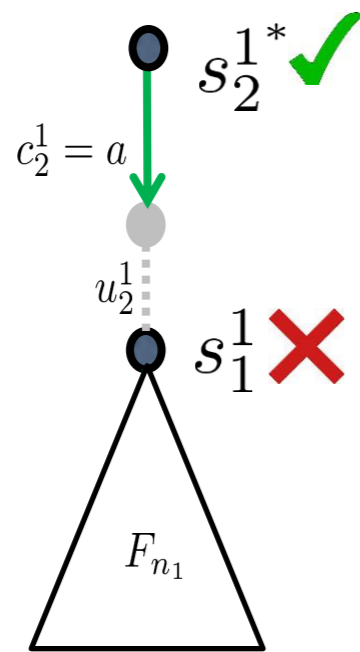
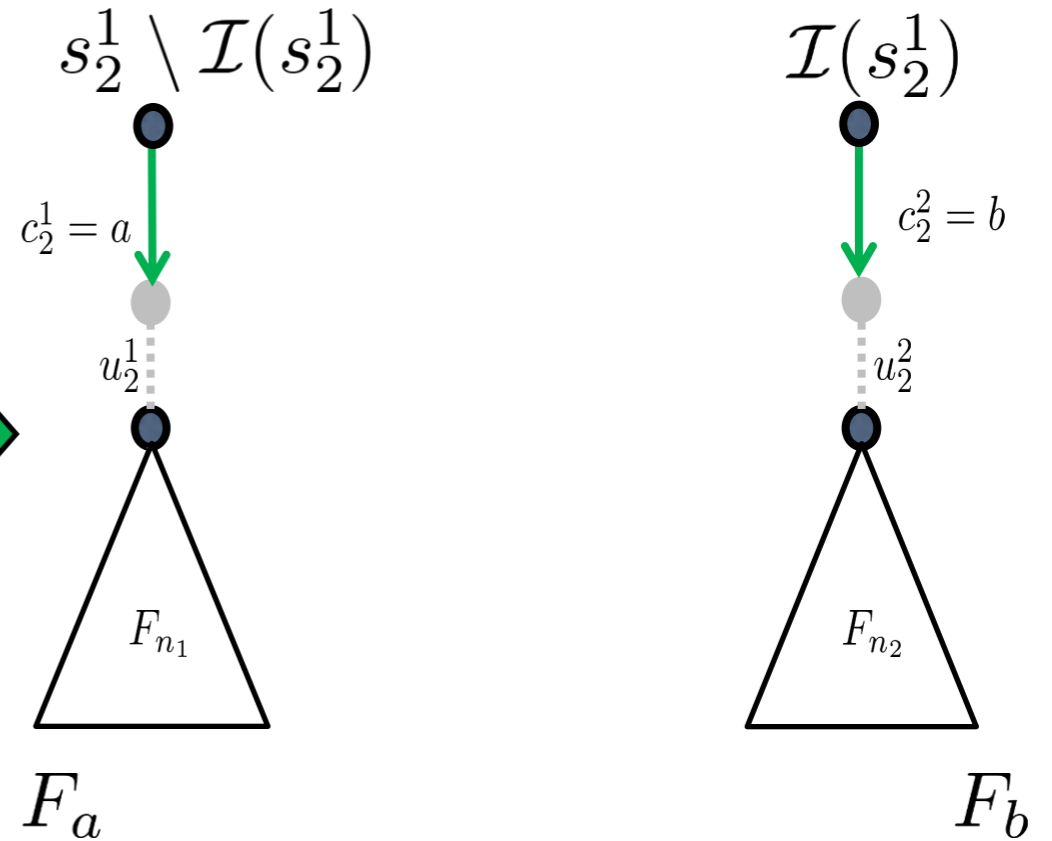
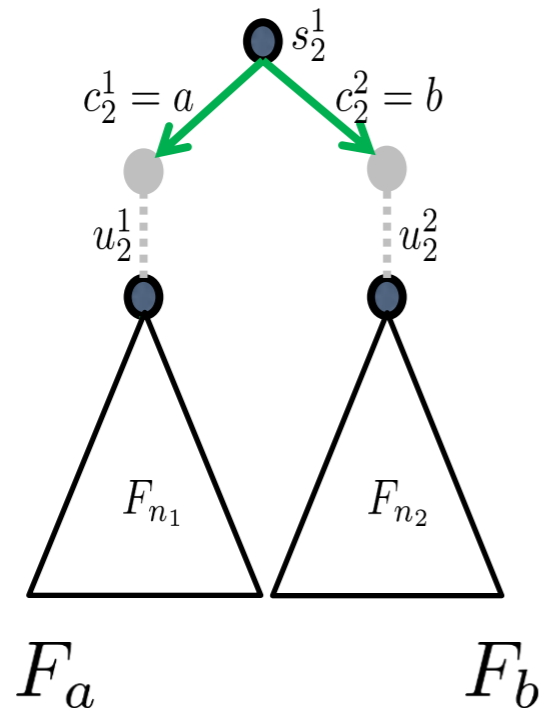
$$T = s_2^{1*} \wedge \delta(s_{n_1}, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

AllSAT(T)

AllSAT(T)| s_1^1



Strategy extraction



$$T = s_2^{1*} \wedge \delta(s_{n_1}, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

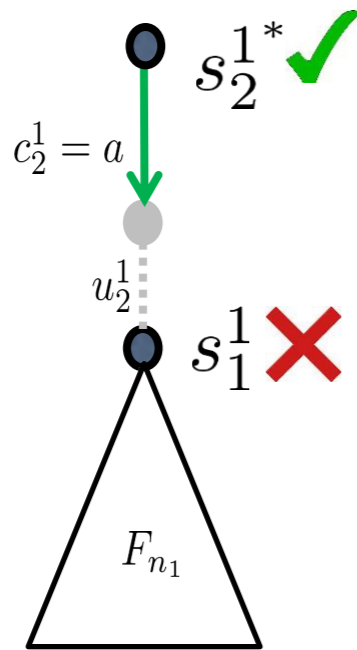
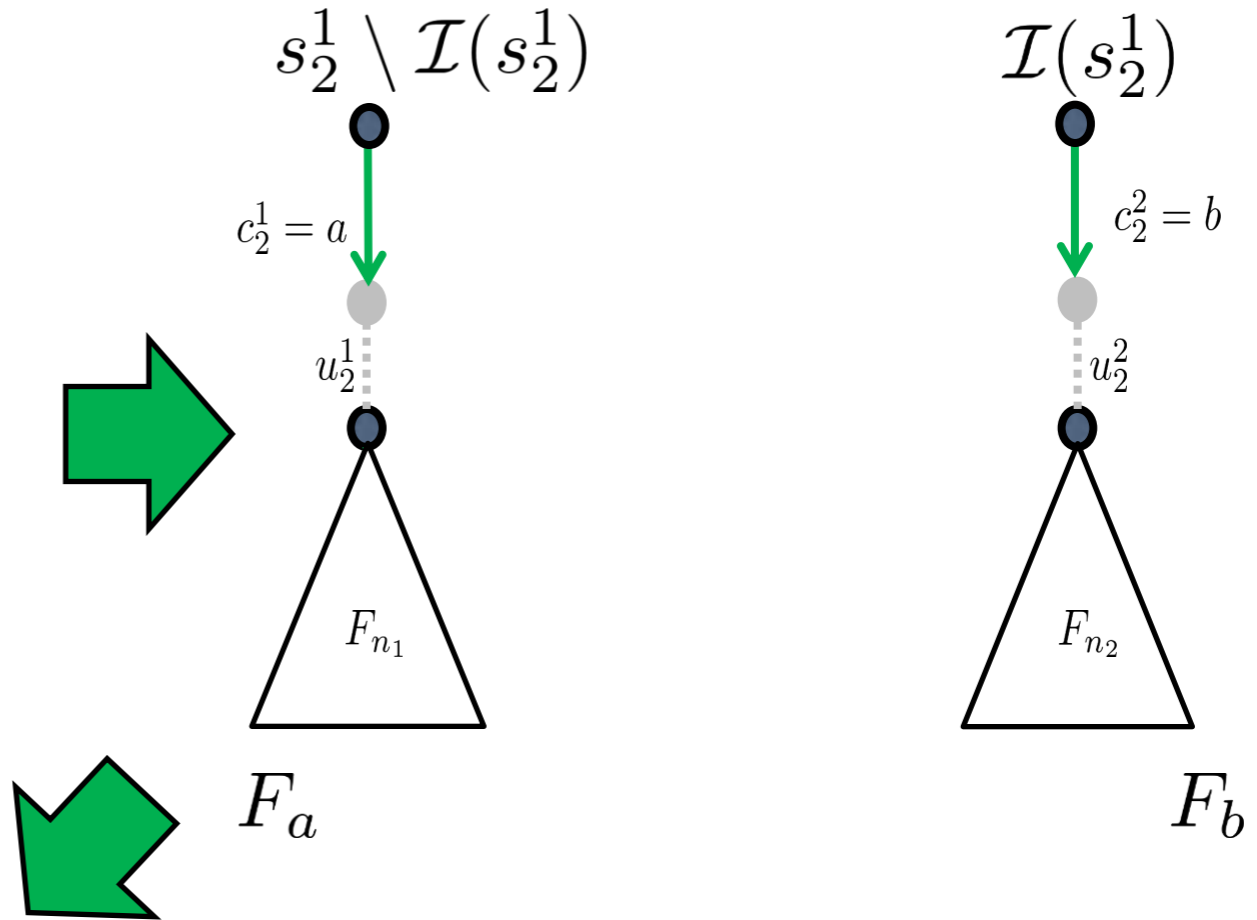
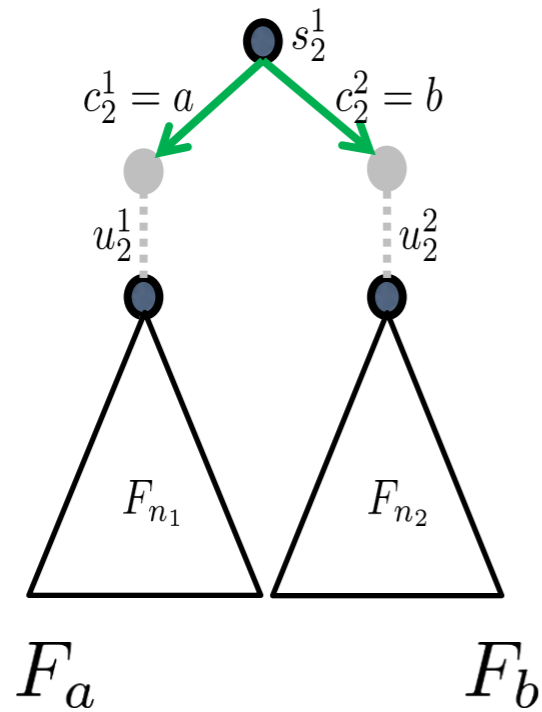
AllSAT(T)

AllSAT(T)| s_1^1

Y. Yu, P. Subramanyan, N. Tsiskaridze, S. Malik:
All-SAT Using Minimal Blocking Clauses, 2014



Strategy extraction



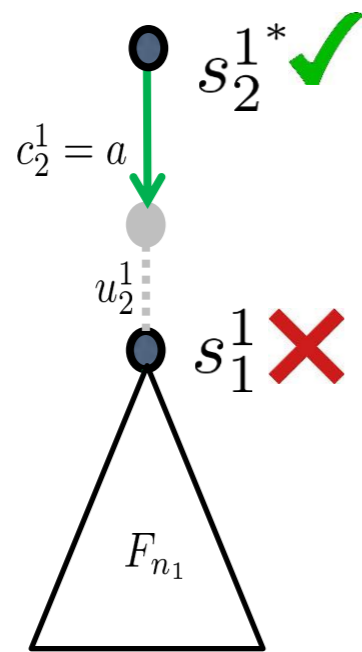
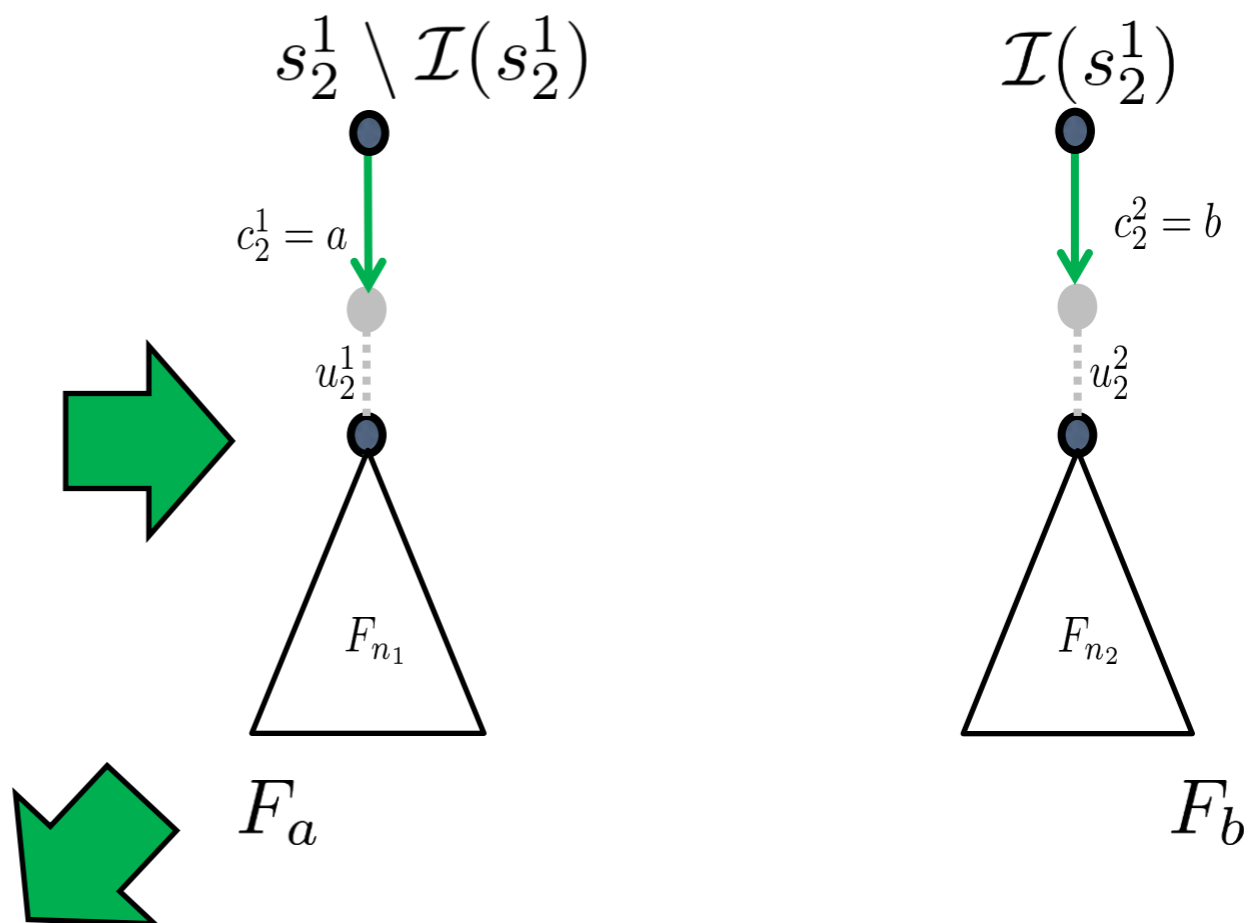
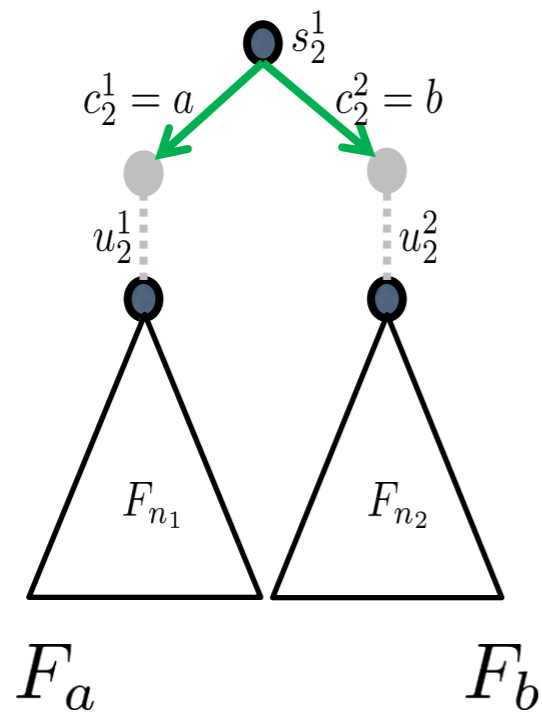
$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1, (c_2^1 = a))$$

AllSAT
 $\text{AllSAT}(T)|_{s_1^1}$

Y. Yu, P. Sukhanyan, N. Tsiskaridze, S. Malik:
 All-SAT Using Minimal Blocking Clauses, 2014



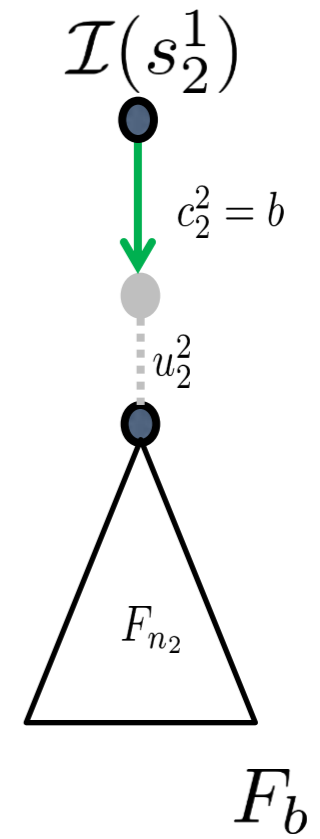
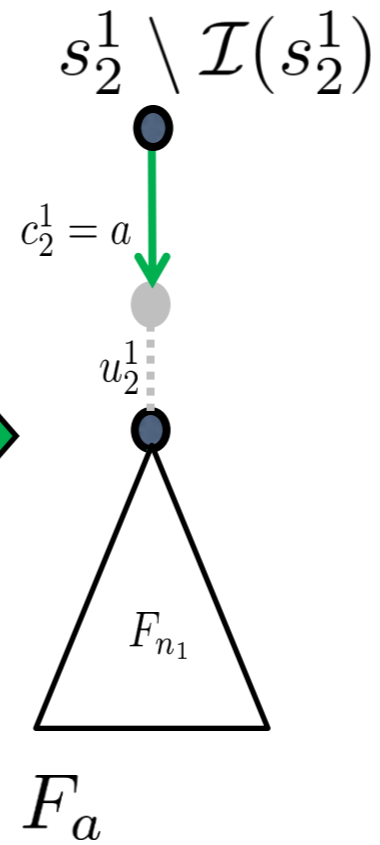
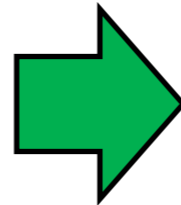
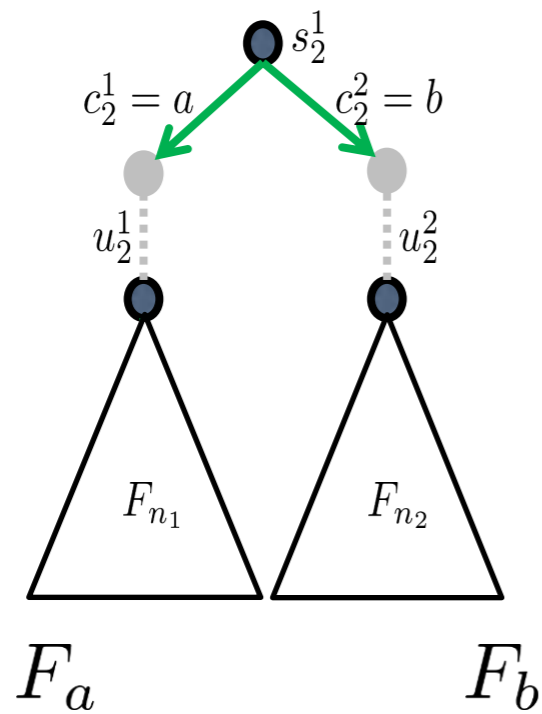
Strategy extraction



$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

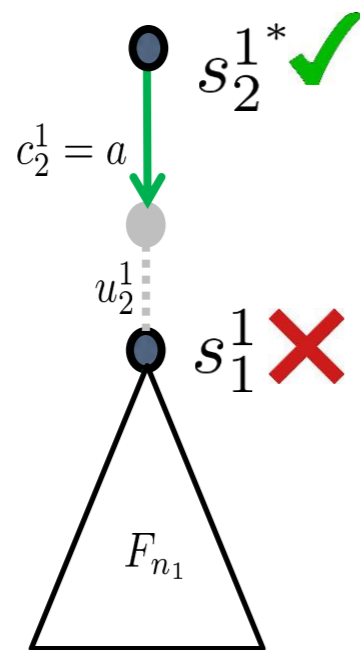


Strategy extraction

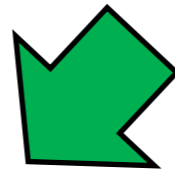
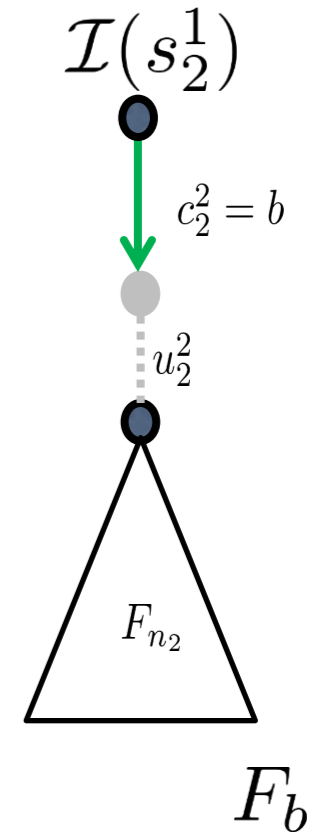
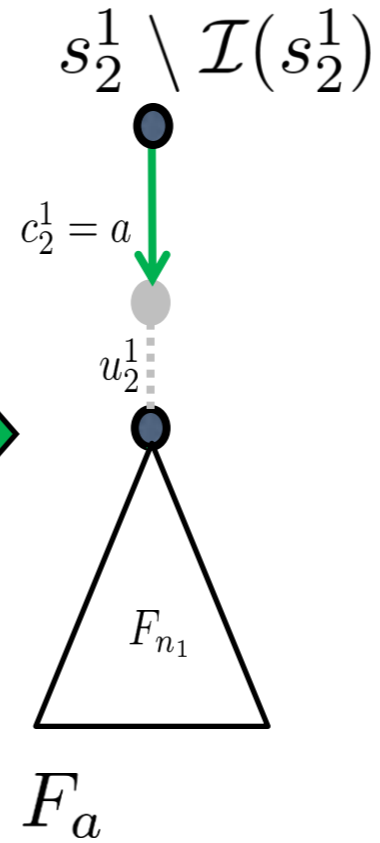
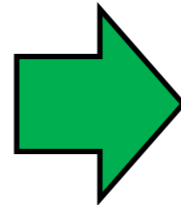
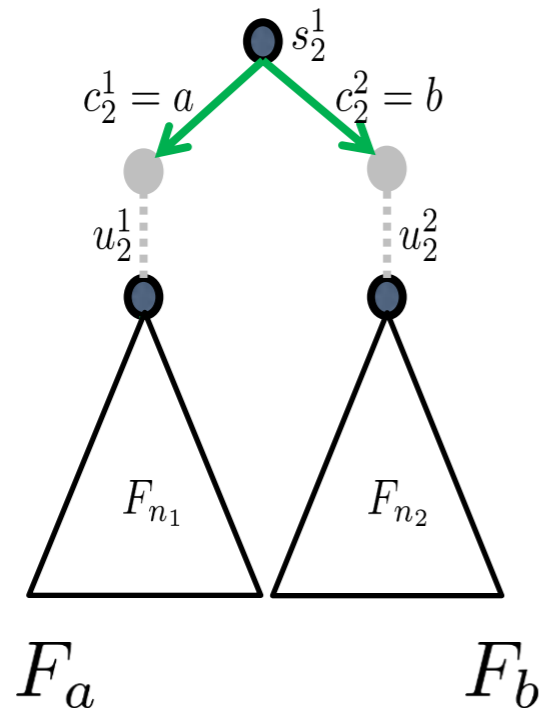


$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

F_{n_1}

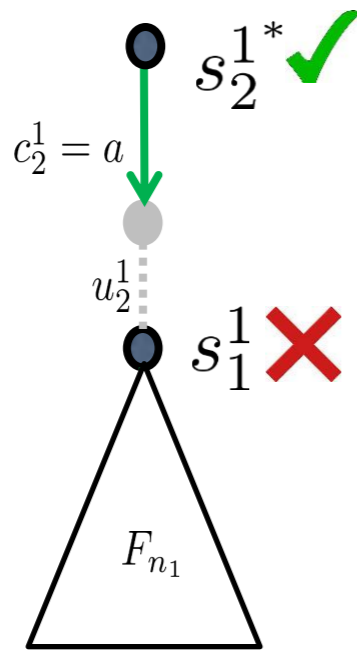


Strategy extraction

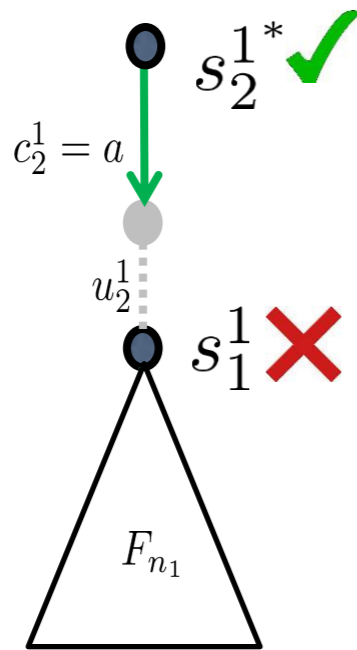
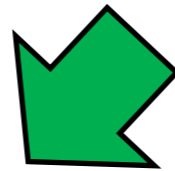
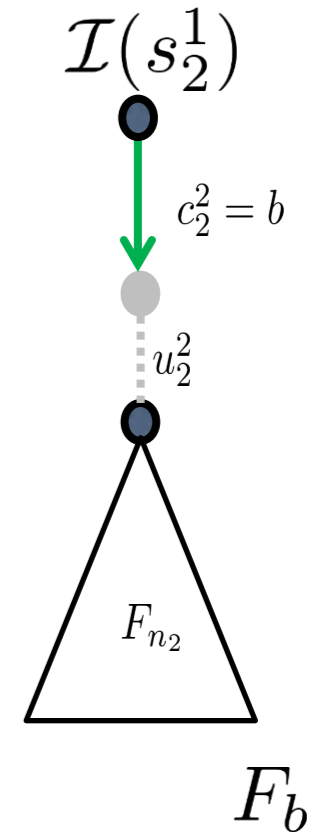
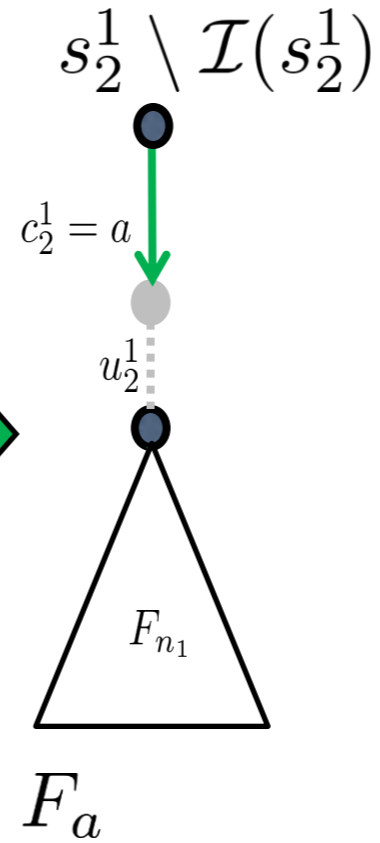
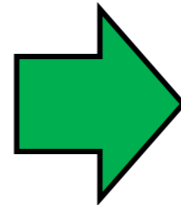
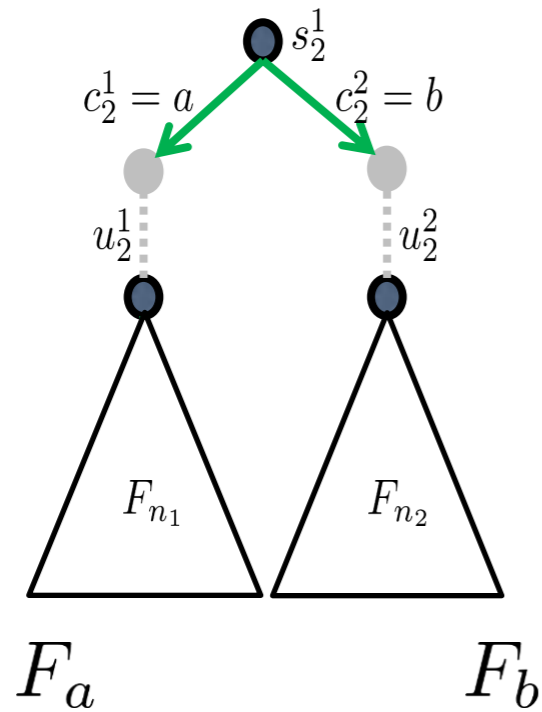


$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

$$F_{n_1} \wedge T = \perp$$



Strategy extraction

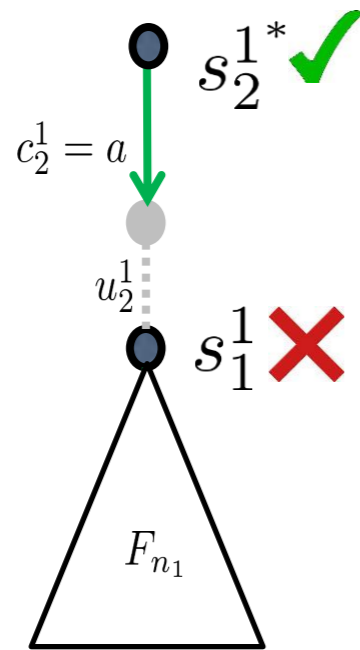
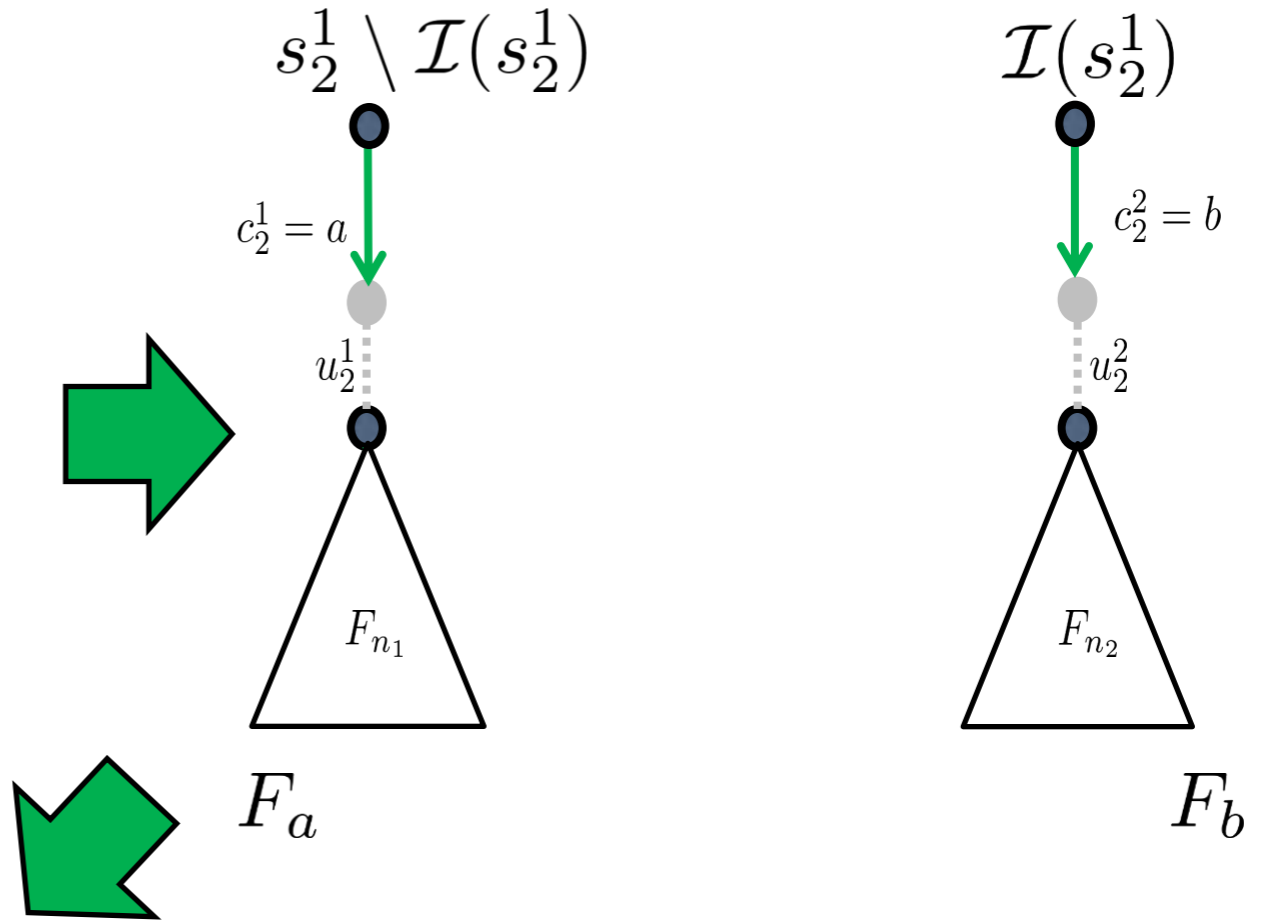
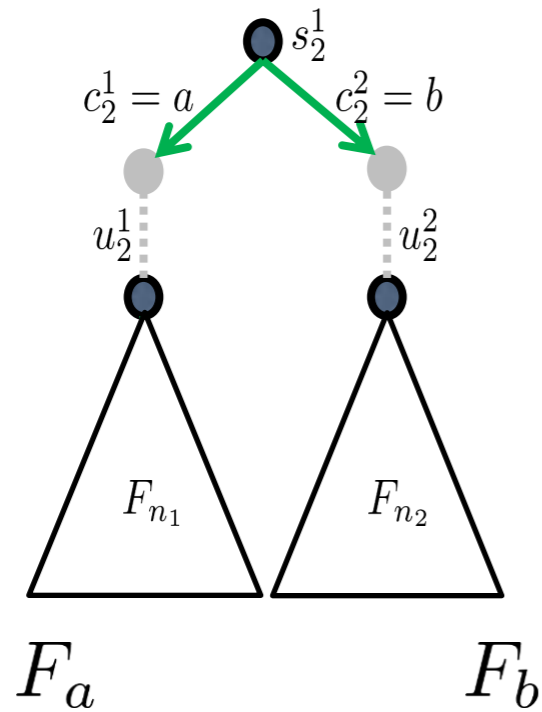


$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

$$F_{n_1} \wedge T = \perp, \text{vars}(F_{n_1}) \cap \text{vars}(T) = s_1^1$$



Strategy extraction



$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

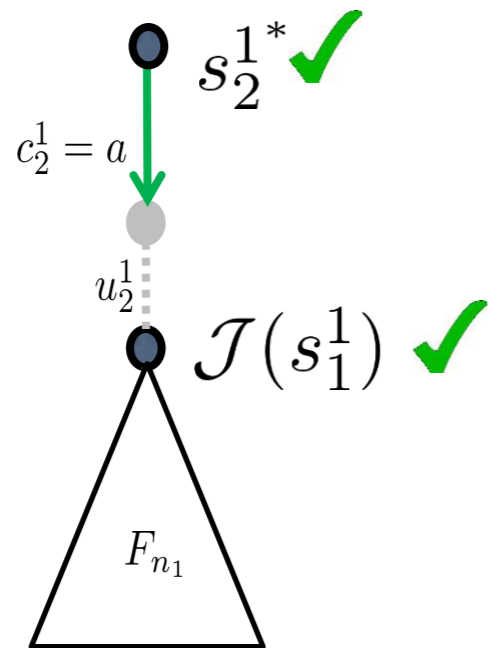
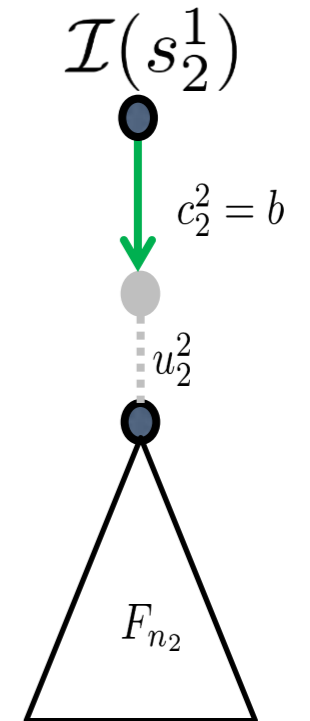
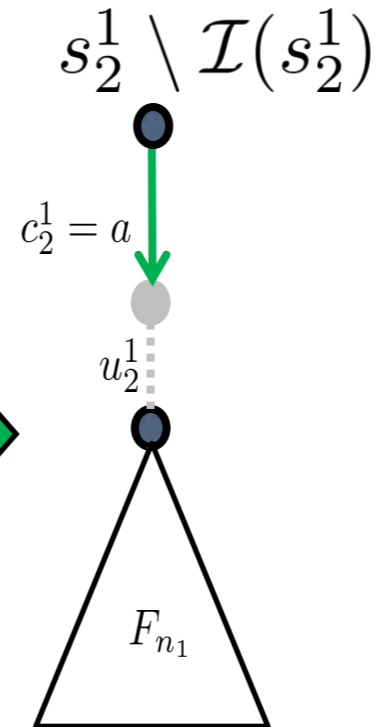
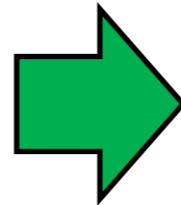
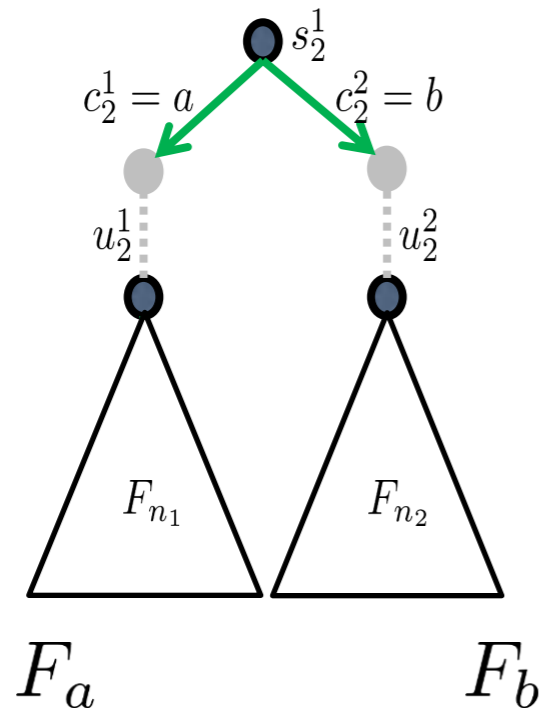
$$F_{n_1} \wedge T = \perp, \text{vars}(F_{n_1}) \cap \text{vars}(T) = s_1^1$$

$$\mathcal{J} = \text{Interpolant}(T, F_{n_1}):$$

- $T \implies \mathcal{J}$
- $\mathcal{J} \wedge F_{n_1} = \perp$



Strategy extraction



$$T = s_2^{1*} \wedge \delta(s_n, c_2^1, u_2^1, s_1^1) \wedge (c_2^1 = a)$$

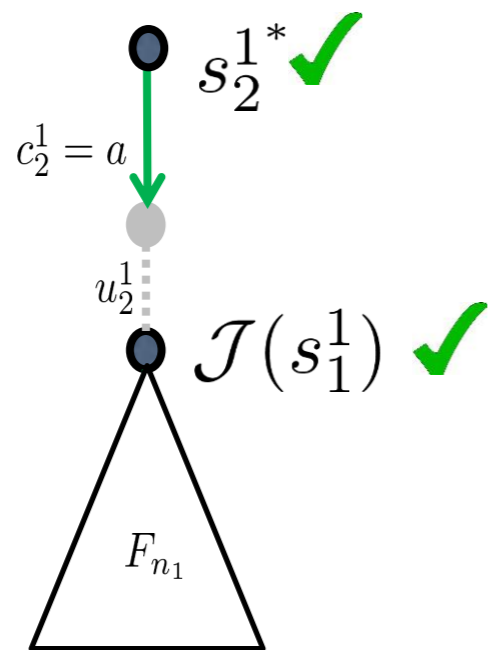
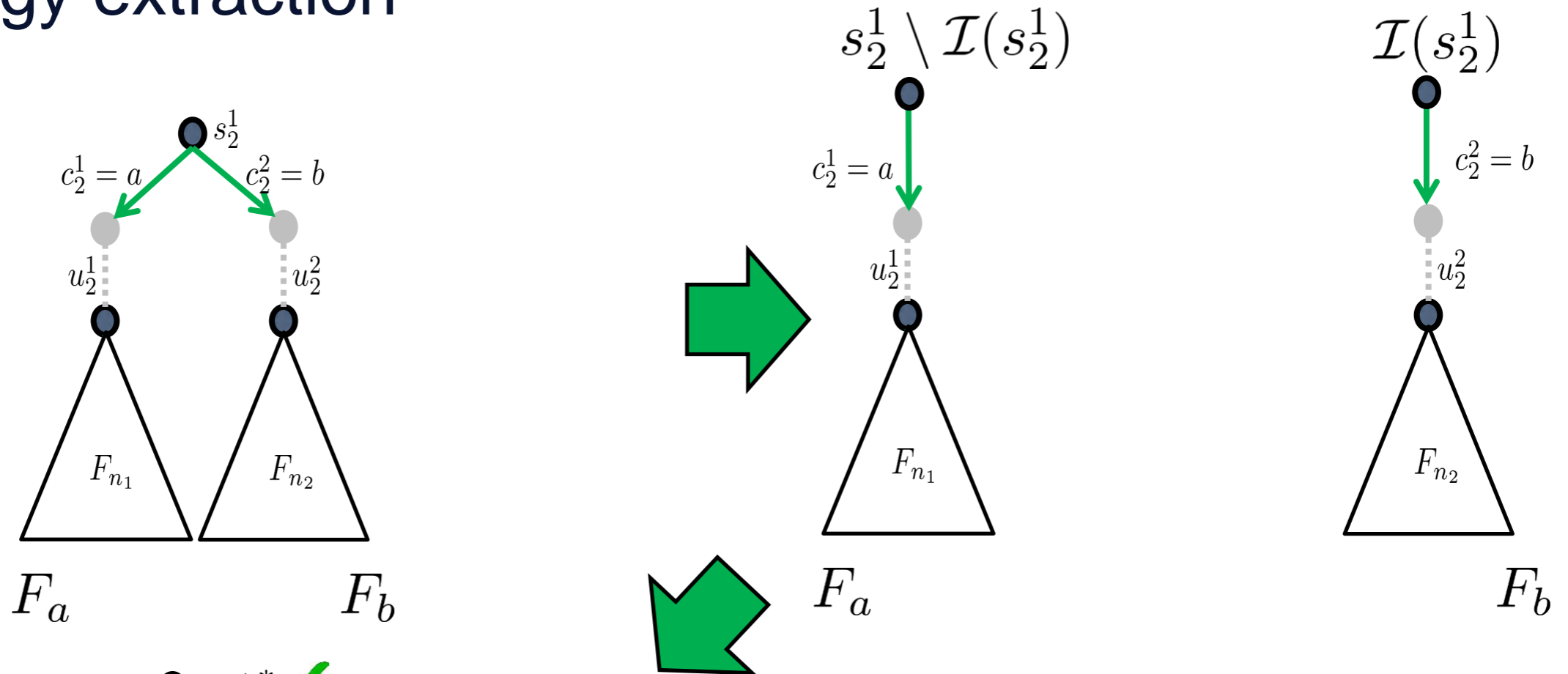
$$F_{n_1} \wedge T = \perp, \text{vars}(F_{n_1}) \cap \text{vars}(T) = s_1^1$$

$\mathcal{J} = \text{Interpolant}(T, F_{n_1})$:

- $T \implies \mathcal{J}$
- $\mathcal{J} \wedge F_{n_1} = \perp$



Strategy extraction



$\mathcal{J}(s_1^1)$ is an overapproximation of reachable states.



Conclusion

We proposed a strategy extraction algorithm based on interpolants

11:15 pm, July 18th, IPRA

Alexander Legg, Nina Narodytska and Leonid Ryzhyk
Practical CNF Interpolants Via BDDs

12:35 pm, July 21st, CAV

N. Narodytska, A. Legg, F. Bacchus, L. Ryzhyk and A. Walker
Solving Games without Controllable Predecessor

14:50 pm, July 21st, CAV

P. Cerny, T. Henzinger, A. Radhakrishna, L. Ryzhyk and T. Tarrach
Regression-free Synthesis for Concurrency

09:00 am, July 24th, SYNT

Leonid Ryzhyk

Automatic Device Driver Synthesis Project (Invited Talk, OSDI'14)



Thanks!

Questions?

