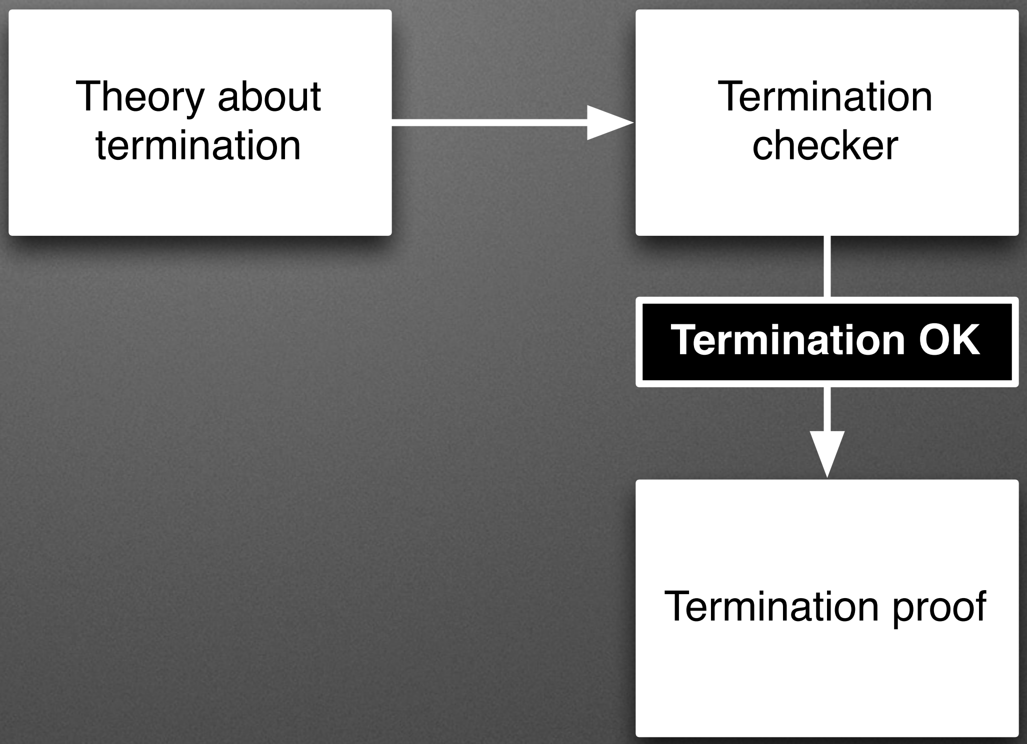
Certification of Termination for Integer Transition Systems

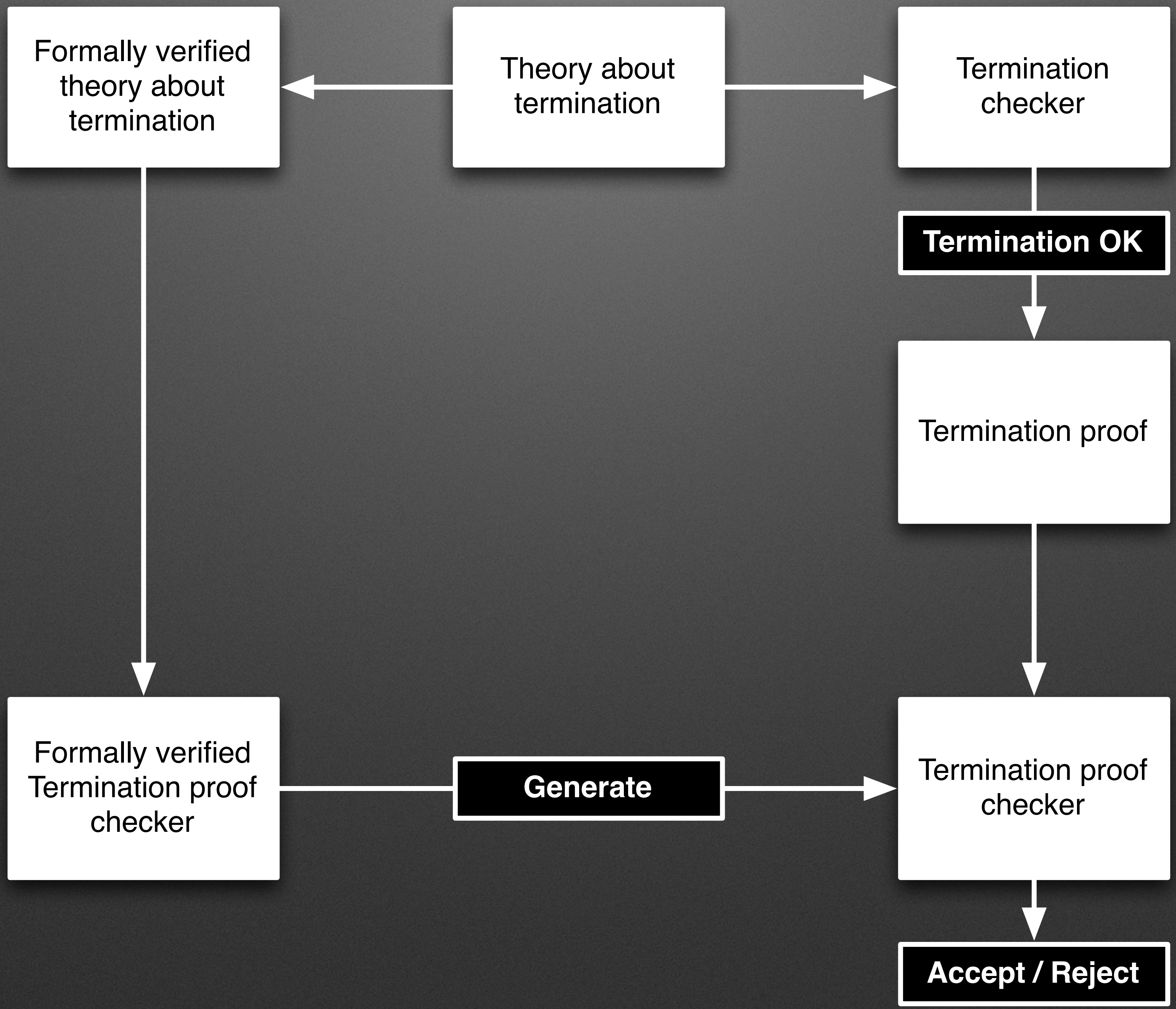
Marc Brockschmidt, *Sebastiaan Joosten*, René Thiemann and Akihisa Yamada
Sebastiaan.Joosten@uibk.ac.at

Supported by FWF project Y 757

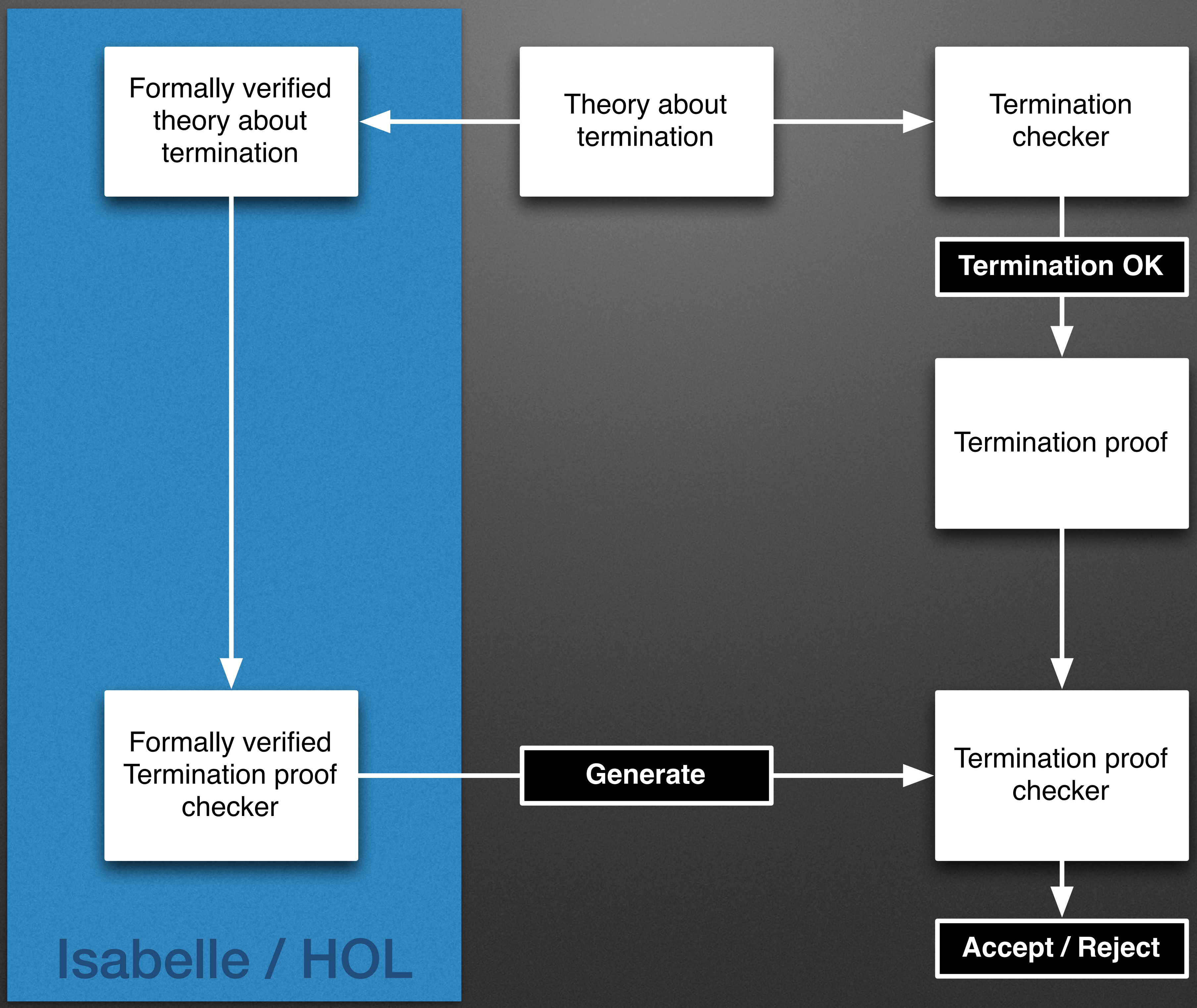
Reliable software



Reliable software



Reliable software



Outline

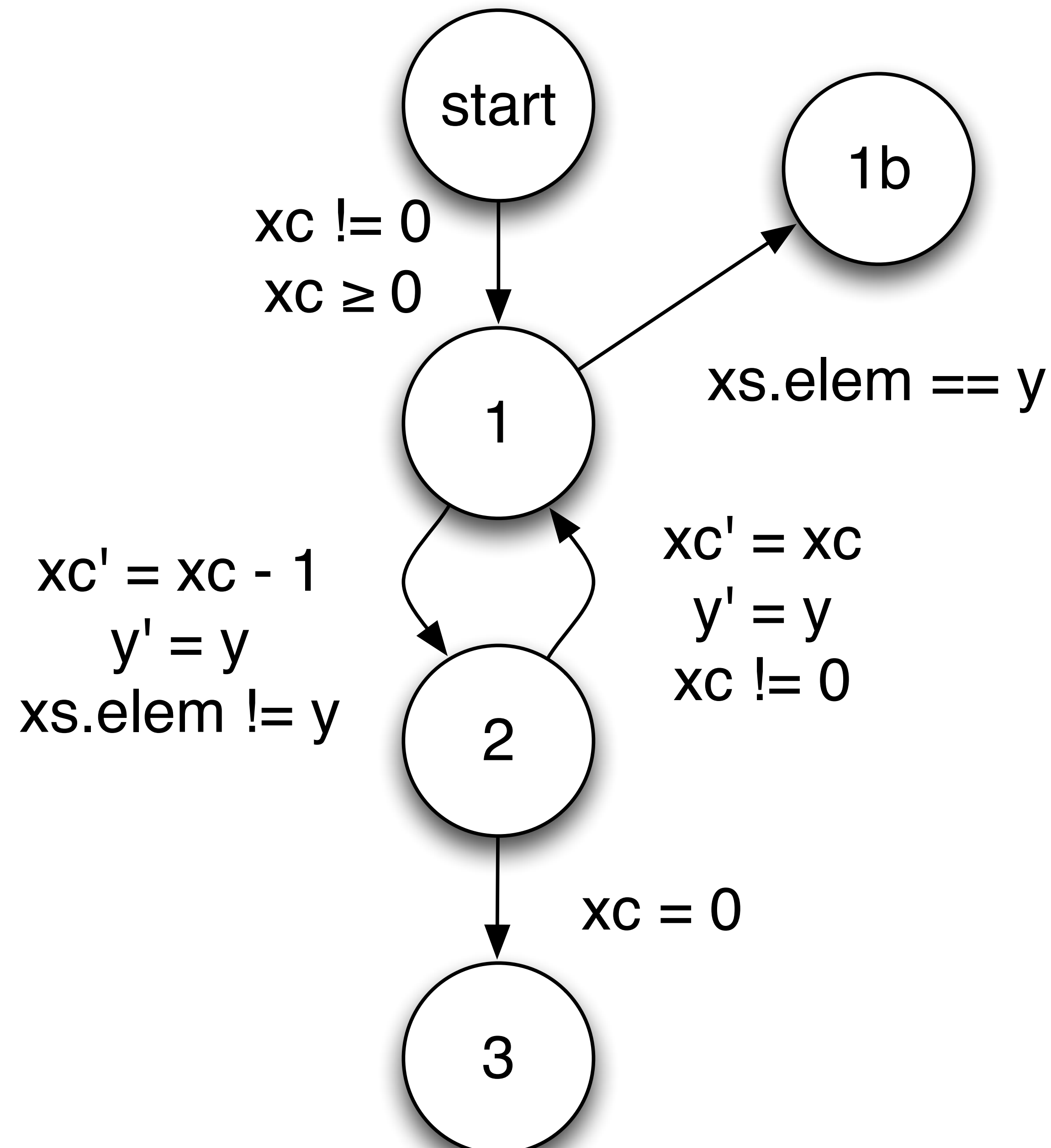
- How to assure termination?
  - Program as labeled transition system
  - Finding termination arguments
- What is essential in the proof of termination?
- What else can we check?

Program as labeled transition system (LTS)

- Place in the program => Label
- Variable update => Transition condition
- Termination => No infinite path from the start node

Program as LTS (example)

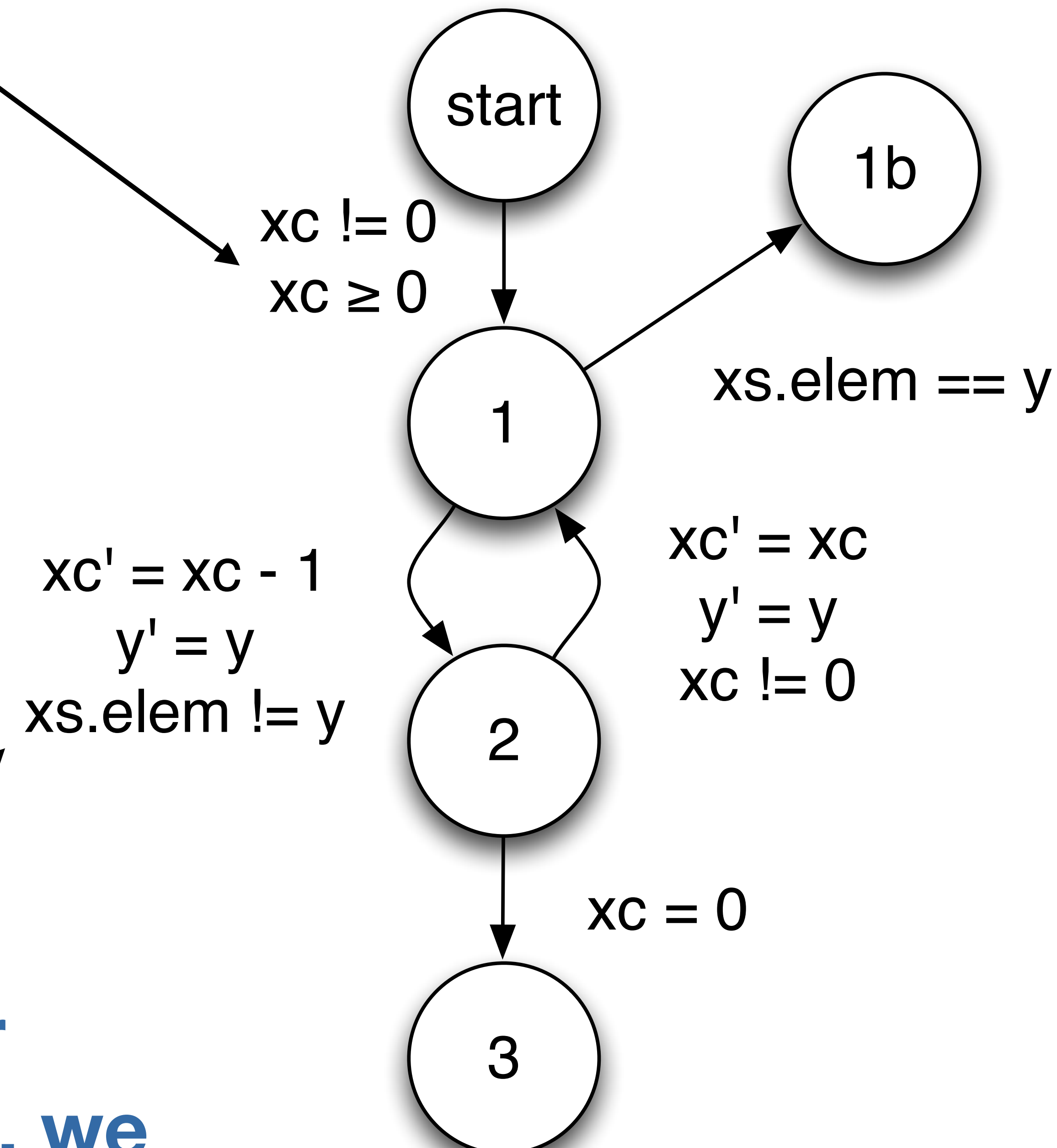
```
function contains(List xs, y){
   while (xs != null){
     if (xs.elem.equals(y))
                      return true;
     xs = xs.next;
   } return false;
}
```



Program as LTS (example)

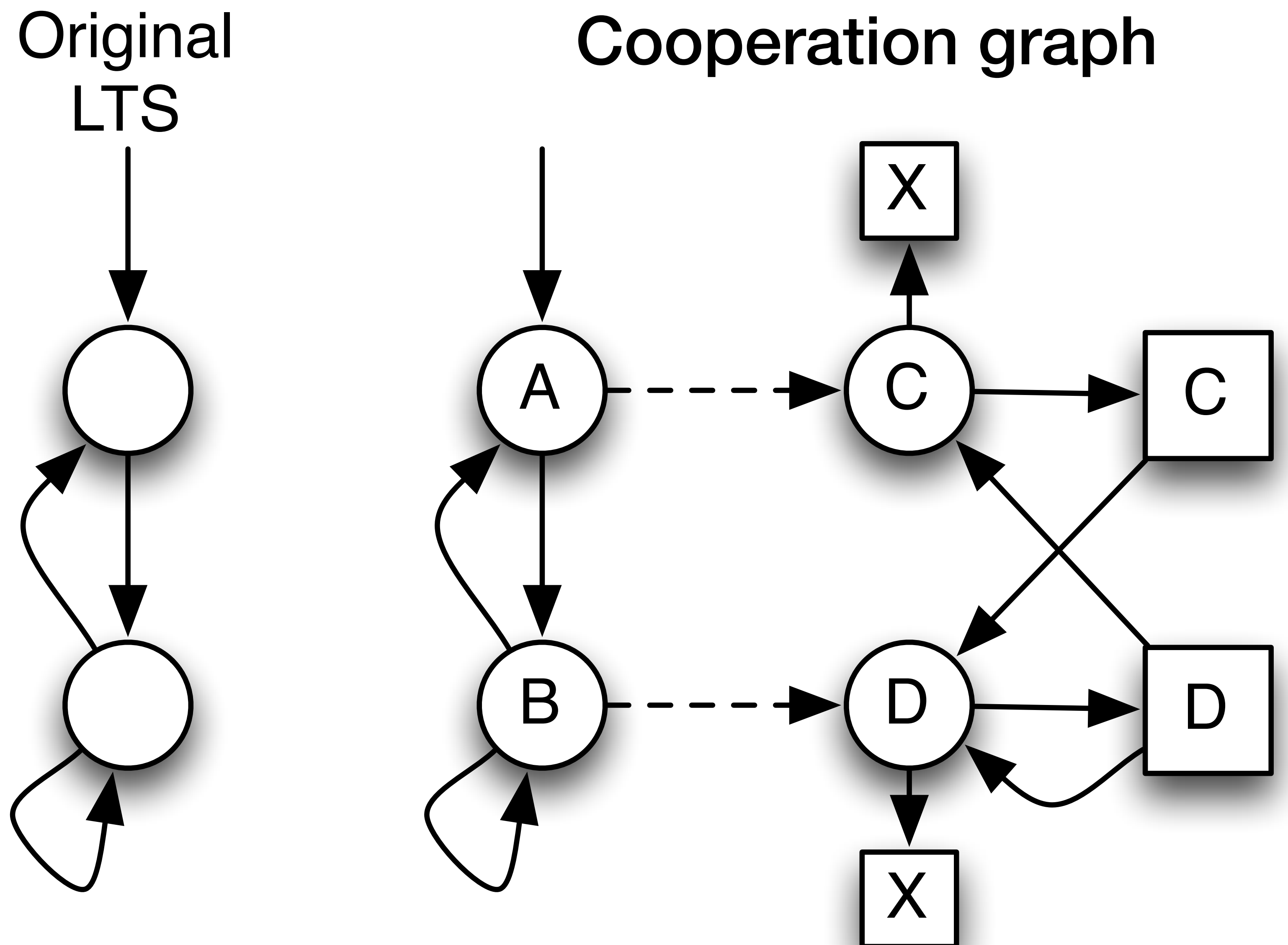
Conditional termination can be encoded as extra conditions

- ```
  function contains(List xs, y){
    while (xs != null){
      if (xs.elem.equals(y))
                      return true;
        xs = xs.next;
    } return false;
  }
  ```



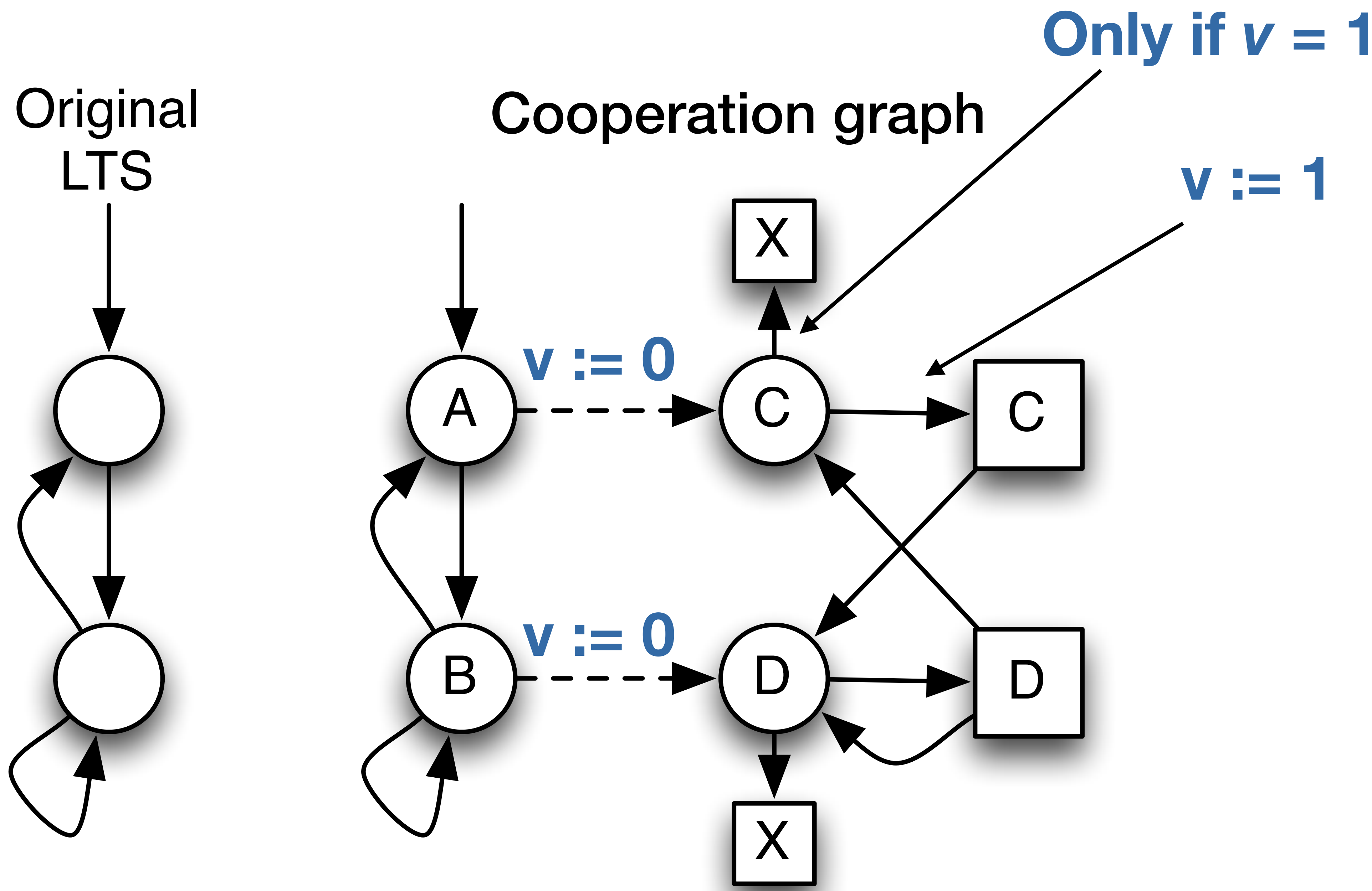
If all constraints are conjunctions of linear inequalities over Integers, we call it an ITS
  ```

Termination of LTS



Better termination proving through cooperation
Marc Brockschmidt, Byron Cook, Carsten Fuhs

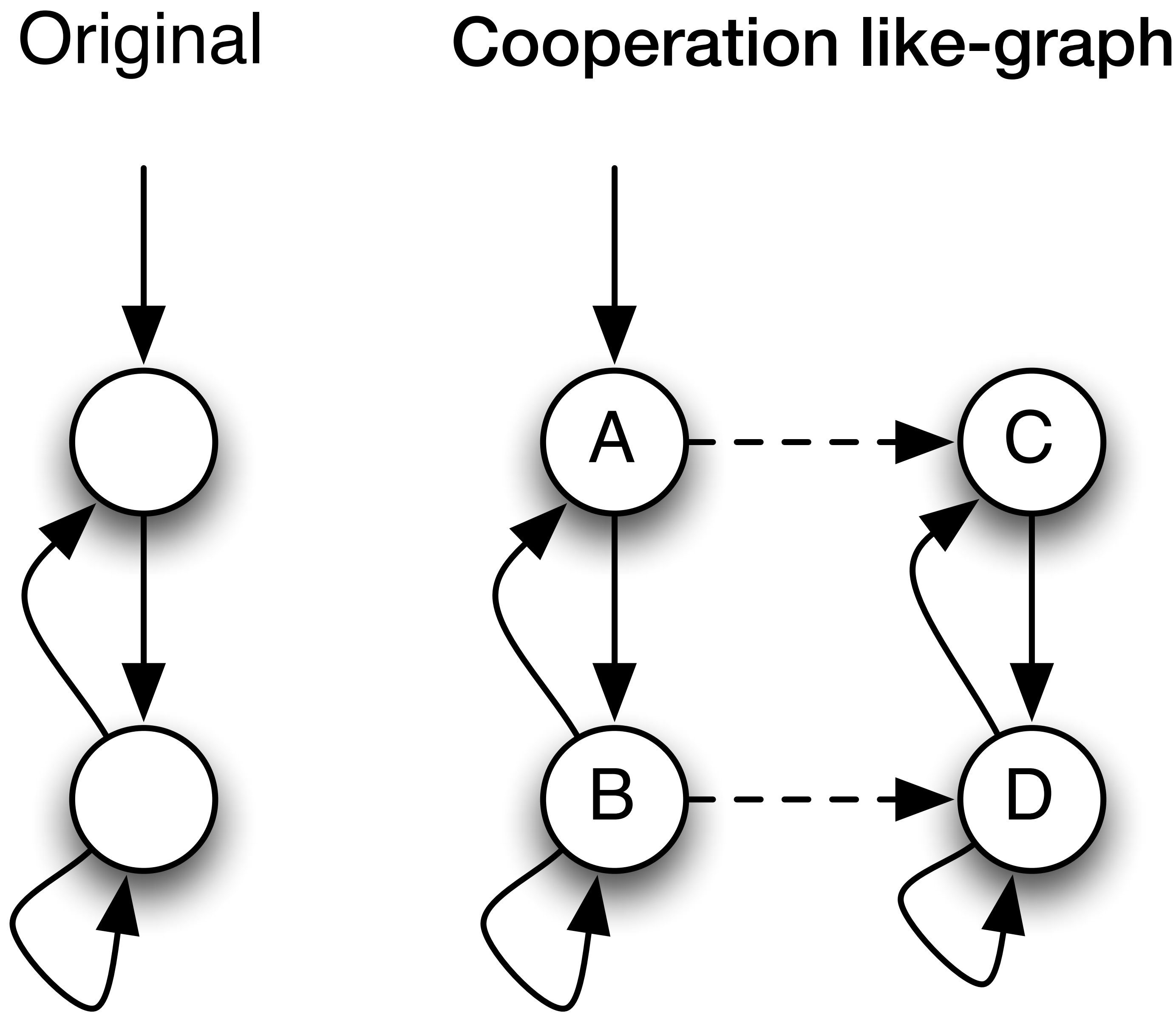
Termination of LTS



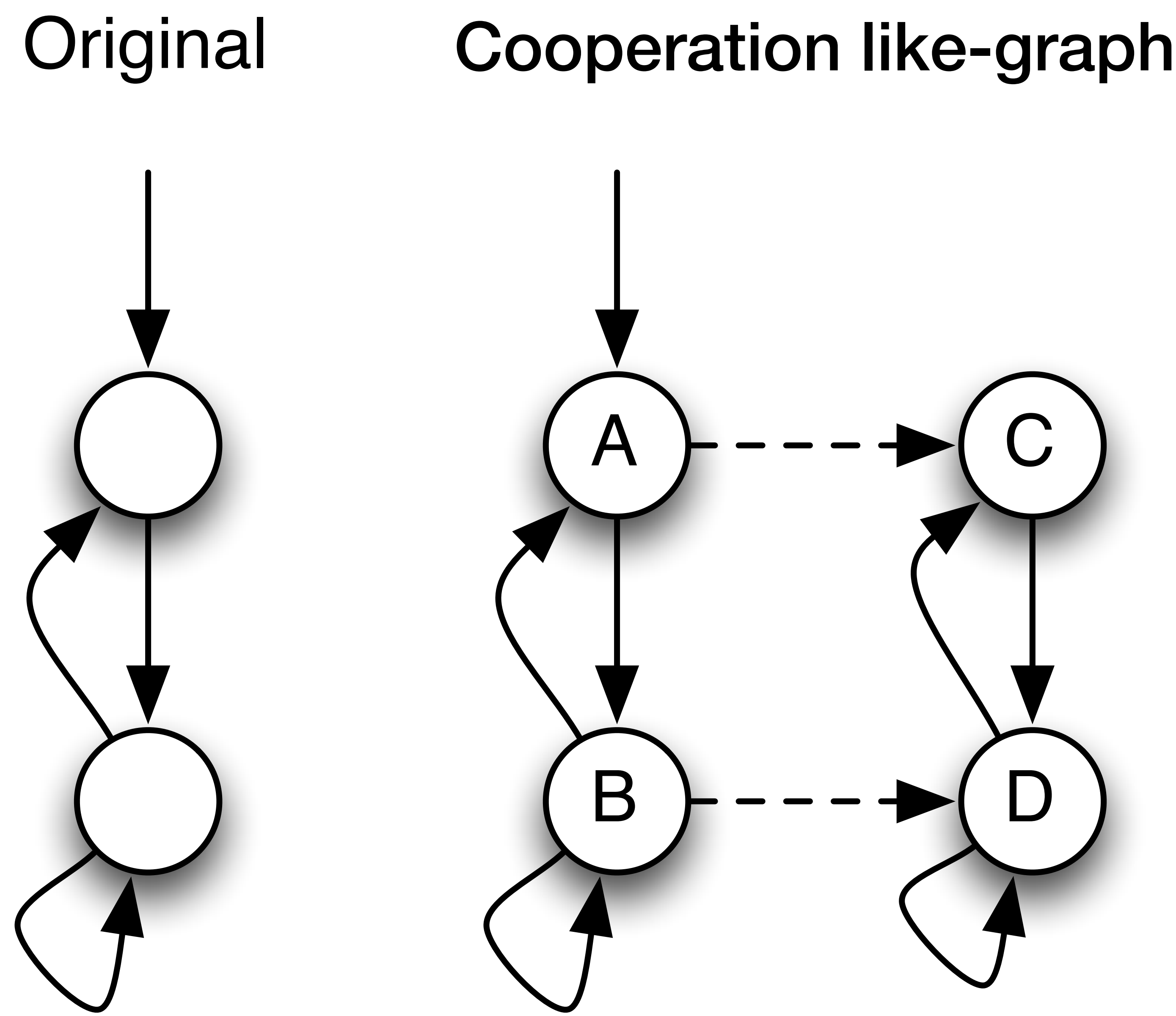
Result:

- argument why the cycle cannot occur infinitely often
- updated Cooperation graph

Termination-proof of LTS



Infinite path through original
=>
Infinite path through cooperation graph



Termination-proof of LTS

```
lemma initial_cooperation_program:
  assumes fin: "finite (transitions R)"
      and copy: "copy_prog R"
      and lts: "lts R"
      and SN: "cooperation_SN P"
  shows "lts_termination R"
```

finite (transitions R) $\Longrightarrow$

copy_prog T P R $\Longrightarrow$

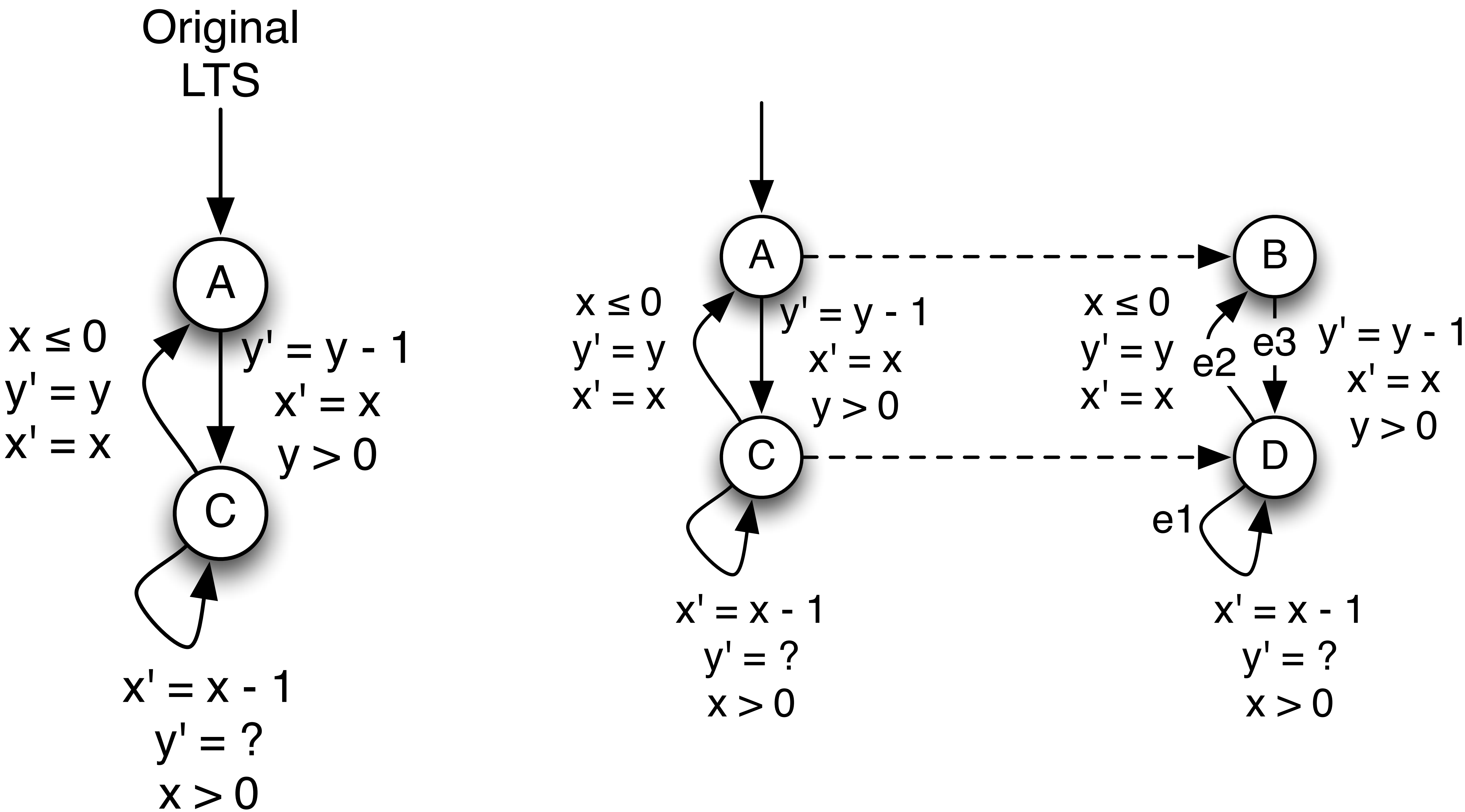
lts T R $\Longrightarrow$

cooperation_SN T P $\Longrightarrow$ lts_termination T R

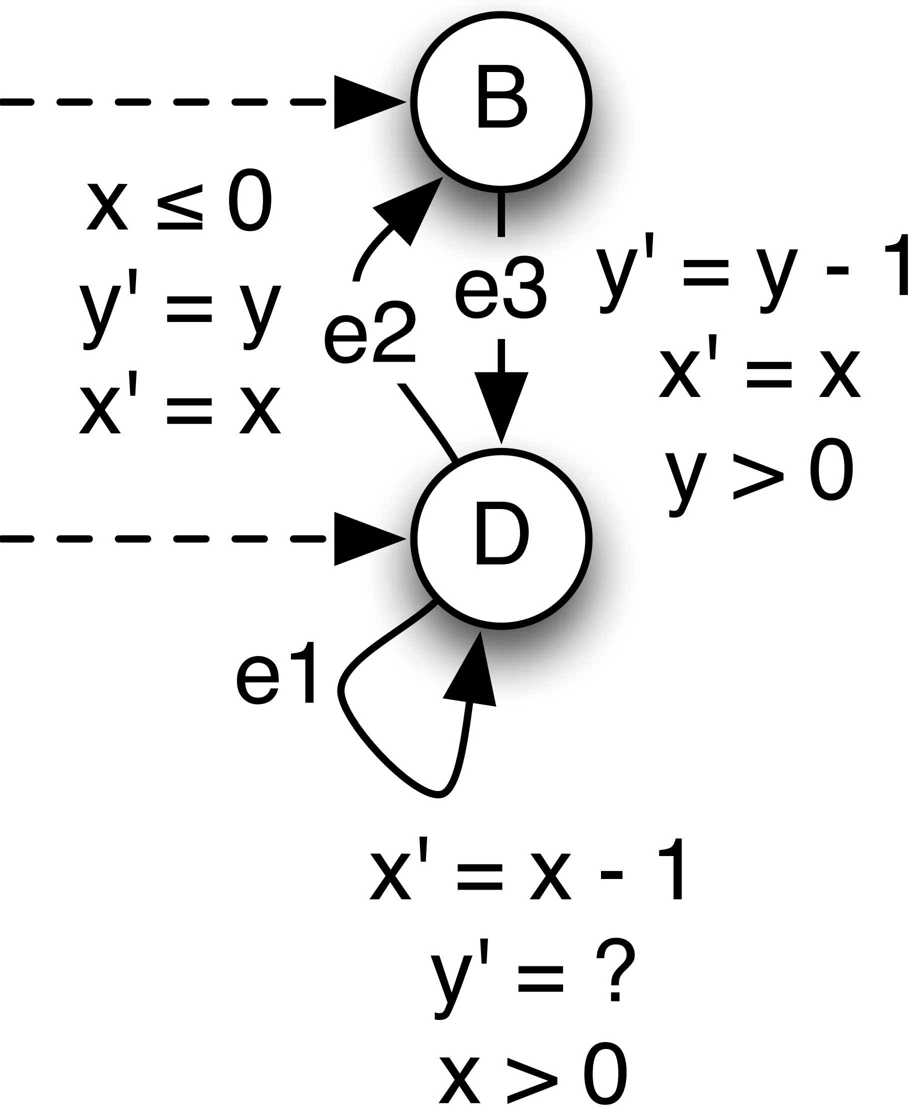
Termination-proof of LTS

- Give a function f(l,a)
  - l: label
  - a: assignment of variables in state with label l
- f is non-increasing for all (remaining) transitions
- f is decreasing for to-be-deleted transitions
- there is a bound, i.e. f cannot decrease infinitely often

Termination-proof of LTS

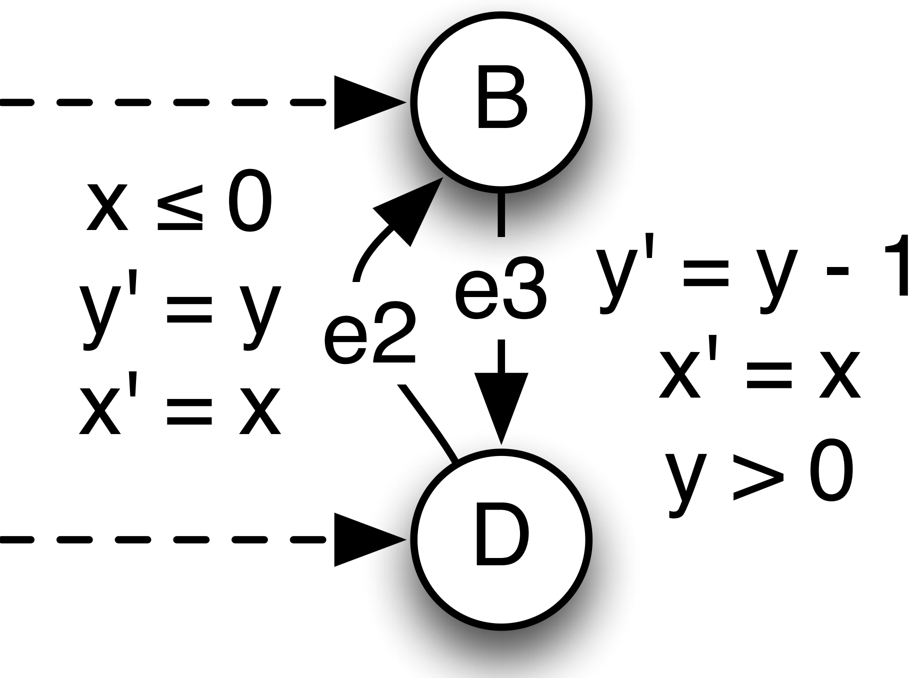


Termination-proof of LTS



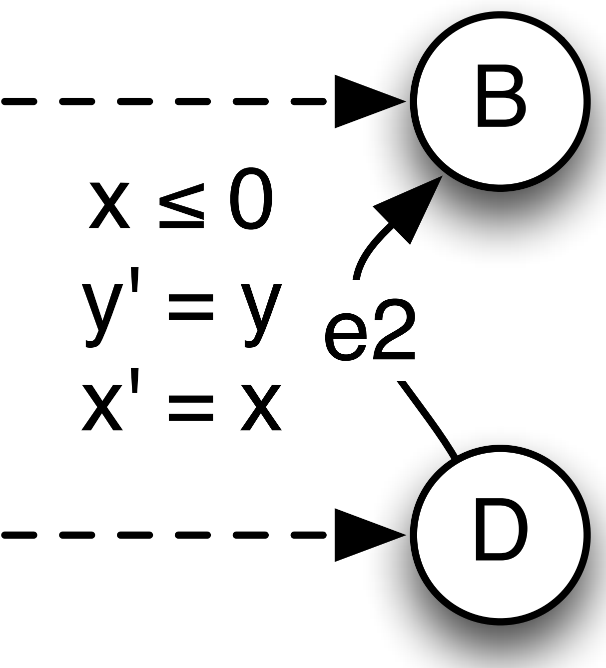
- f(B,x,y,z) = x
  f(D,x,y,z) = x
- e2 and e3 preserve this value.
  Proof: use x’ = x
  e1 strictly decreases it
  Proof: use x’ = x - 1
- Consequently, e1 cannot occur infinitely often (can be removed)

Termination-proof of LTS



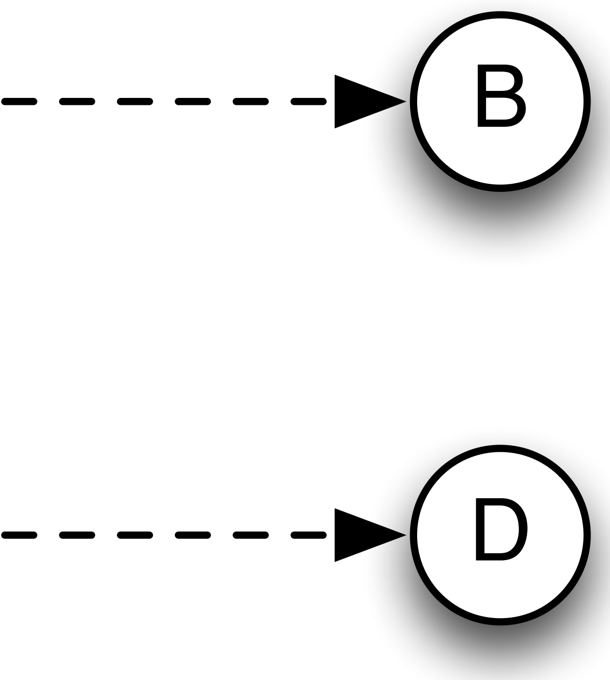
- $f(B,x,y,z) = y$
  $f(D,x,y,z) = y$
- e2 preserves this value.
  Proof: use $y' = y$
  e3 strictly decreases it
  Proof: use $y' = y - 1$
- Consequently, e3 cannot occur infinitely often (can be removed)

Termination-proof of LTS



- f(B,x,y,z) = 1
  f(D,x,y,z) = 2
- e2 strictly decreases the value
  Proof: use no equalities
- Consequently, e2 cannot occur infinitely often (can be removed)

Termination-proof of LTS

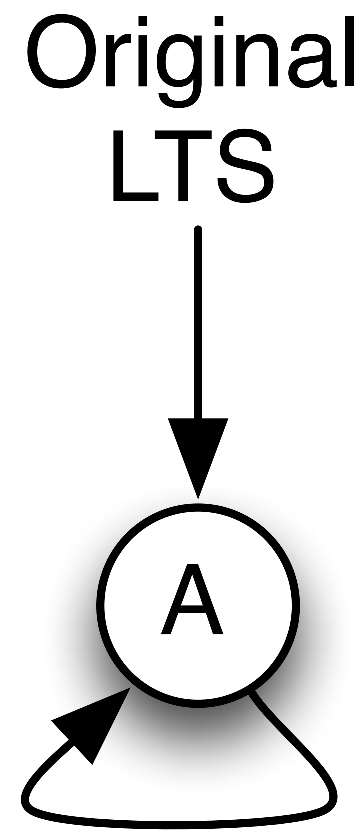


- No transitions can be taken infinitely often
- Program terminates

Strengthening proofs

- Use invariants
- Split nodes
- Add helper-variables
- Use of Lexicographic order in the decrease-function

Termination-proof of LTS

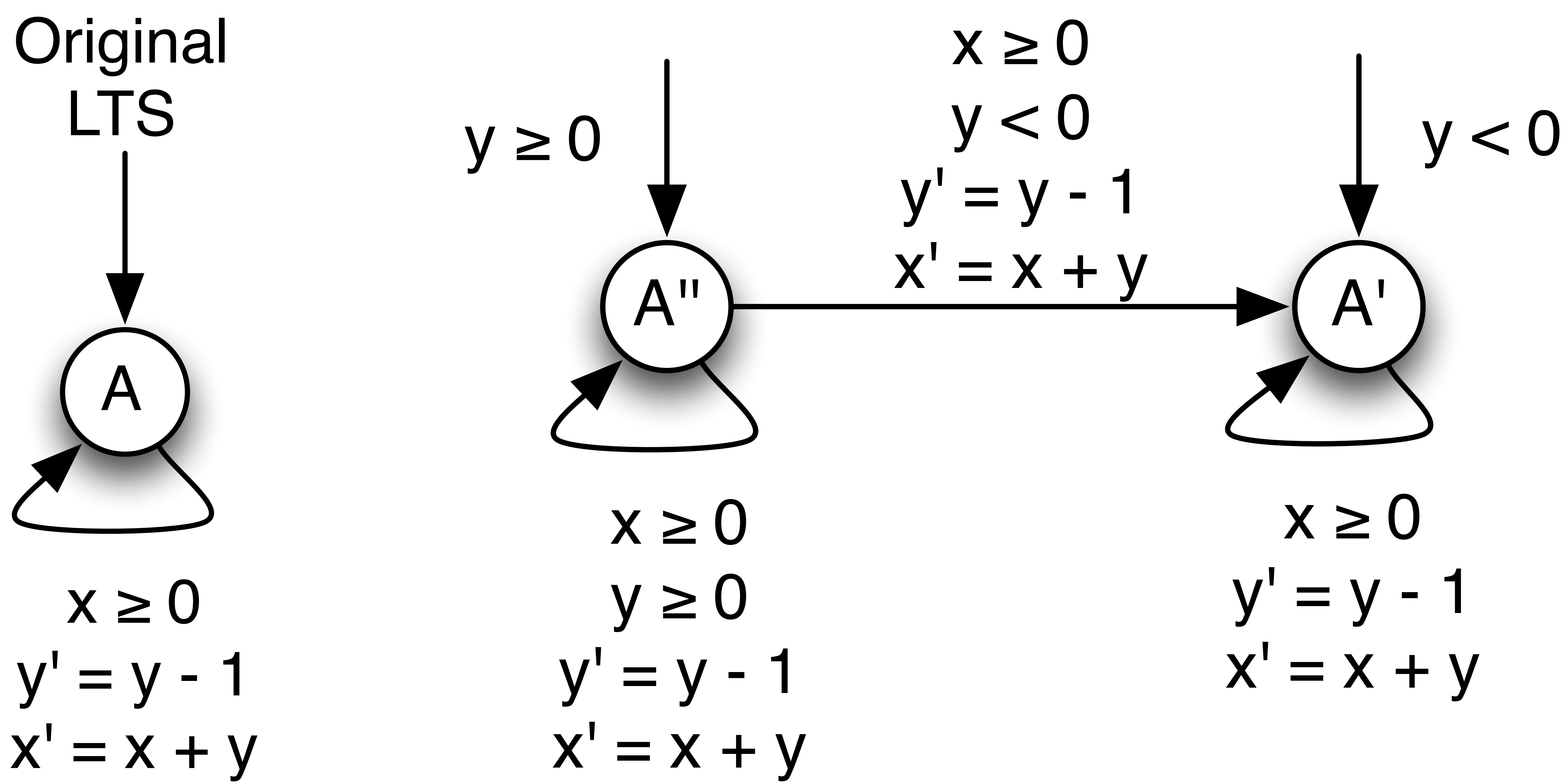


$x \geq 0$

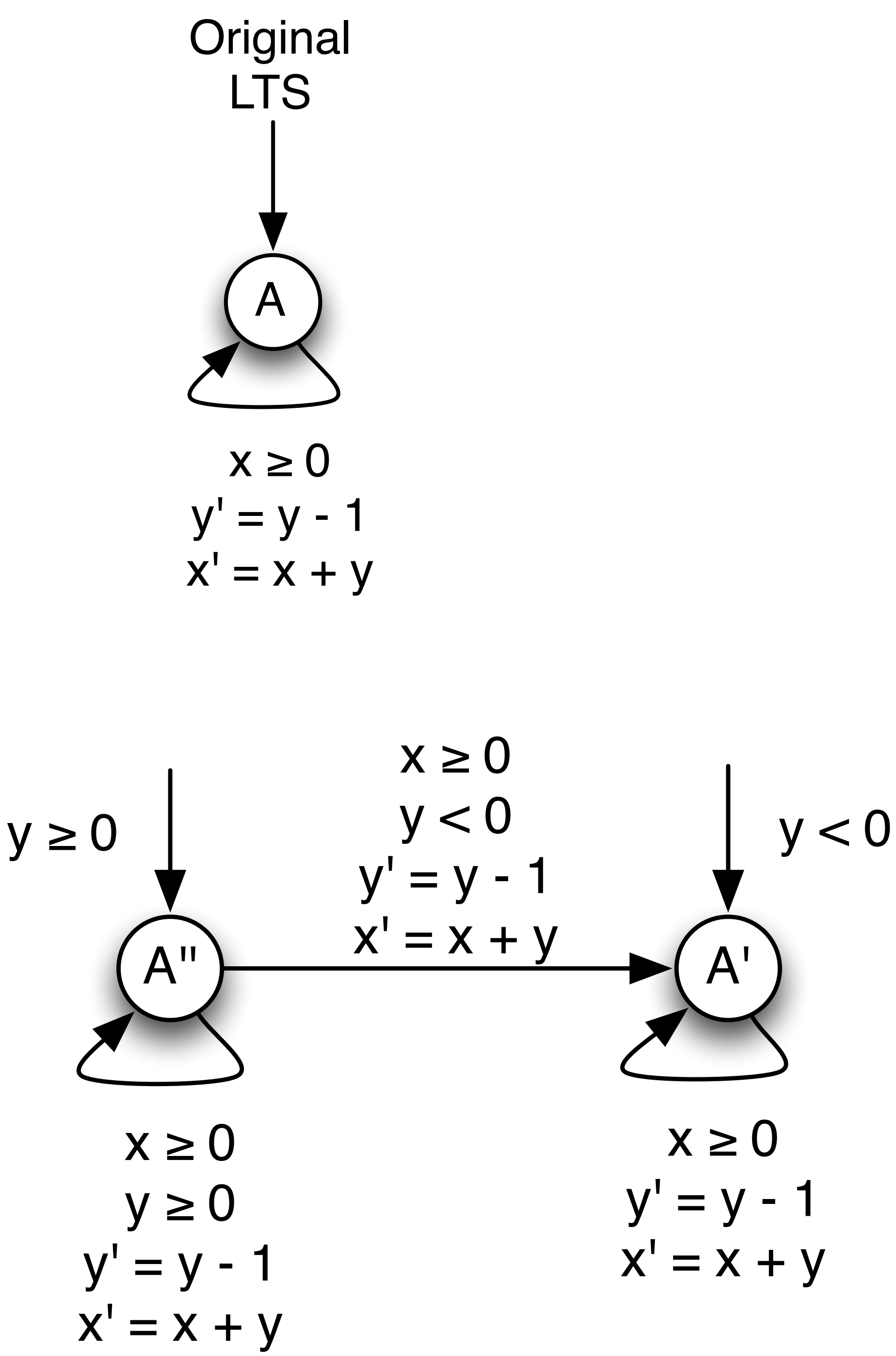
$y' = y - 1$

$x' = x + y$

Termination-proof of LTS



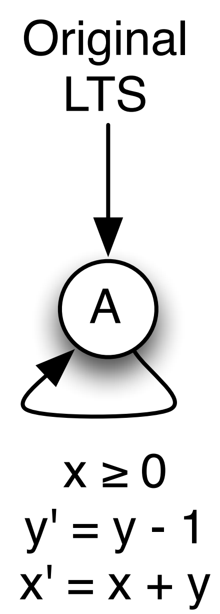
Proof of equivalence

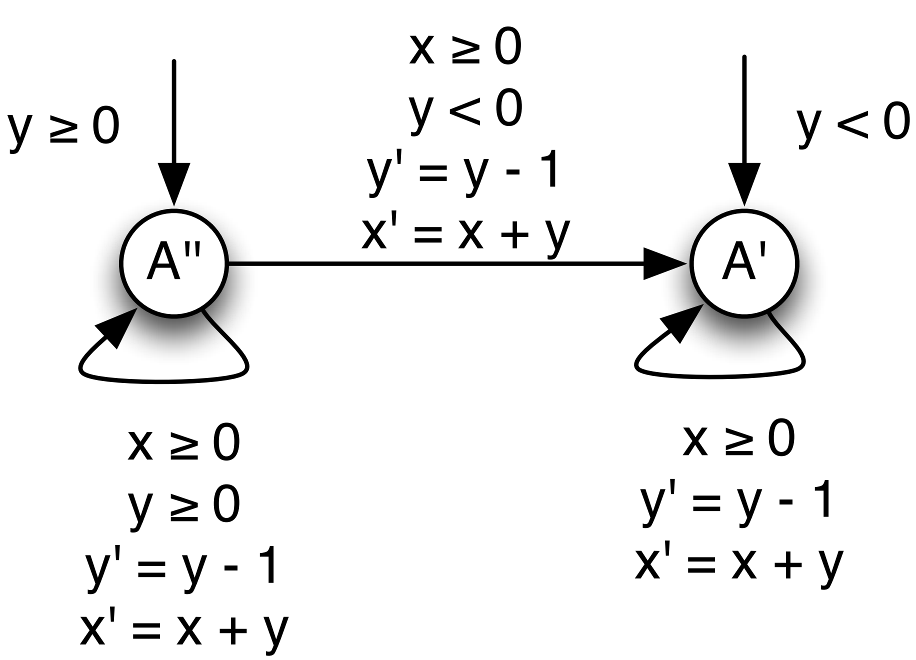


- Initial node:
  True => $y \geq 0 \ \| \ y < 0$
  proof by contradiction:
  $y \leq -1$ && $-y \leq 0$
  adding the inequalities gives:
  $0 \leq -1$

- Similar proofs for the other transitions.

Proof of termination





- In A’, the invariant $y < 0$ holds
- The edge from A’ to A’ can be eliminated through ranking function $f(l,x,y) = x$ (using invariant $y < 0$)
- The edge from A” to A” can be eliminated through $f(l,x,y) = y$
- The edge from A” to A’ can be eliminated through $f(l,x,y) = (l == A’ ? 0 : 1)$

Proof of termination

- State LTS R
- Give an LTS P + invariants for each state
  + a proof that the invariants hold
  + a proof that P can simulate R
  + a proof that cooperation graph P’ terminates
- If P’ has edges: give a function f, a set of edges T
  + a proof for each edge in T that f decreases
  + for other edges in P’: proof that f does not increase
  + a proof of termination for P’ - T

Todo?

- Parser
- Minor theory details
- Anything that comes up during testing

Other properties?

- LTS allows non-determinism
- Checking invariants (“safety checker”) can be used for:
  - Runtime / resource analysis (of linear bounds)

Other theories?

- Formalised currently:
  - Linear integer inequalities
  - Lexicographic combinations of arbitrary existing theories (curently only linear integer inequalities)
- Many other theories can be expressed this way:
  - Booleans as 0/1 integers
  - Bitvectors as lists of Booleans

Other theories?

- Tell us about your proofs!
  sjcjoosten@gmail.com

- Keep an eye out for CeTA 2.28:
  http://cl-informatik.uibk.ac.at/software/ceta/