

Efficient On-the-fly Algorithms for the Analysis of Timed Games

Didier LIME

Center for Embedded Software Systems, Aalborg University, DENMARK

14 avril 2005

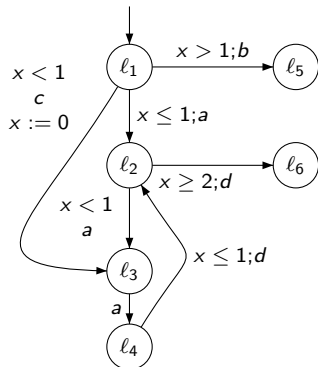
Introduction

- **Controller synthesis** for **timed** systems (e.g. real-time and embedded systems)
 - Given a system S and a property φ find a controller C such that

$$C \parallel S \models \varphi$$

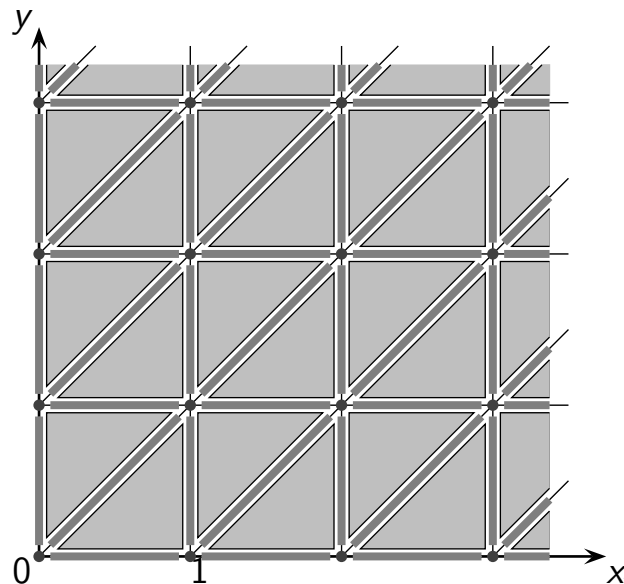
- **Safety** properties
 - **Reachability** properties
- ⇒ Modeled as a 2 player game
- Common work with Kim G. LARSEN, Emmanuel FLEURY, Alexandre DAVID (Aalborg University - CISS) and Franck CASSEZ (IRCCyN, Nantes)

Plan

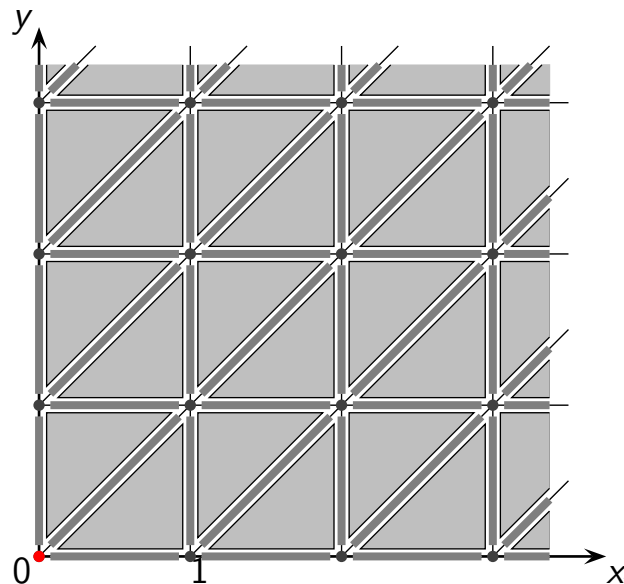


Example (Semantics)

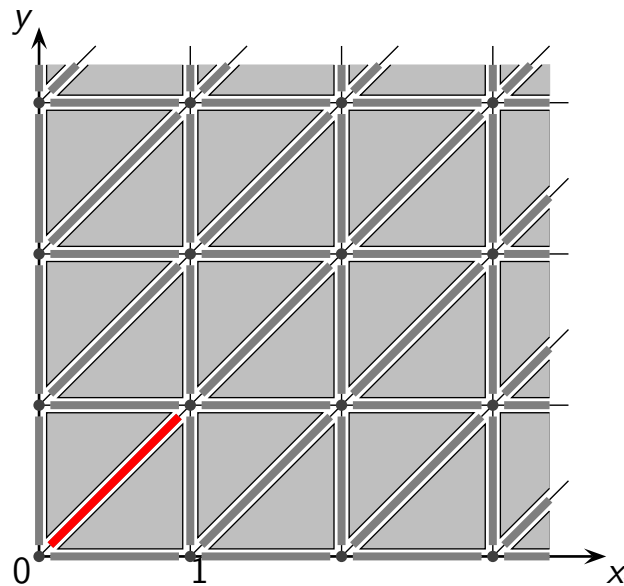
$$\begin{aligned}
 (l_1, 0) &\xrightarrow{0.6} (l_1, 0.6) \xrightarrow{c} (l_3, 0) \xrightarrow{0.39} \\
 (l_3, 0.39) &\xrightarrow{a} (l_4, 0.39) \rightarrow \dots
 \end{aligned}$$



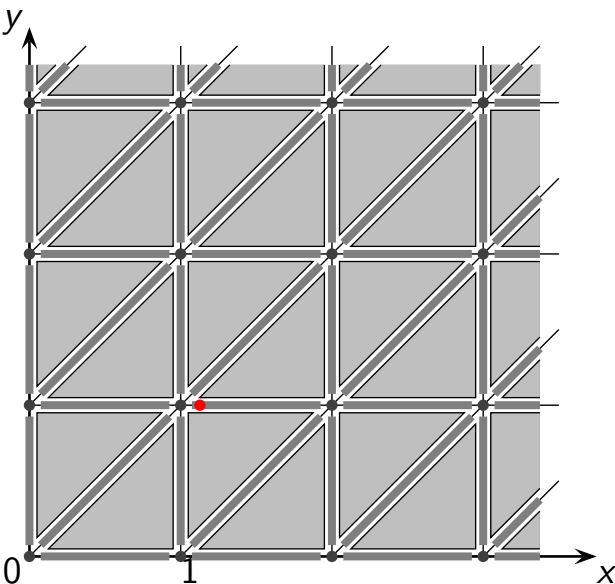
► Skip region graph



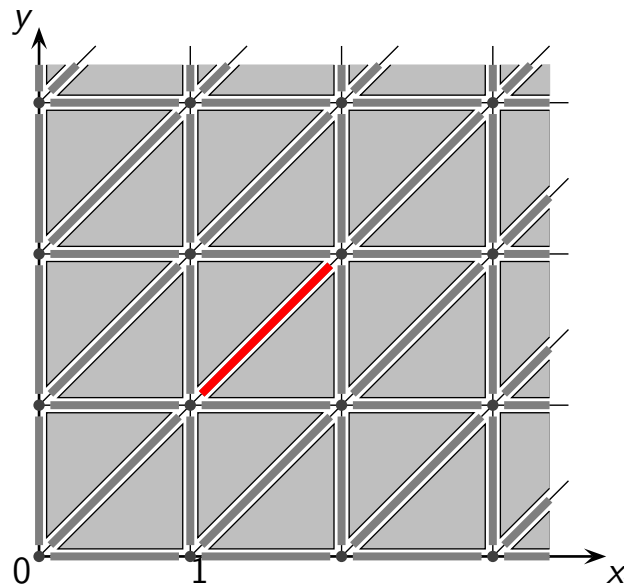
► Skip region graph



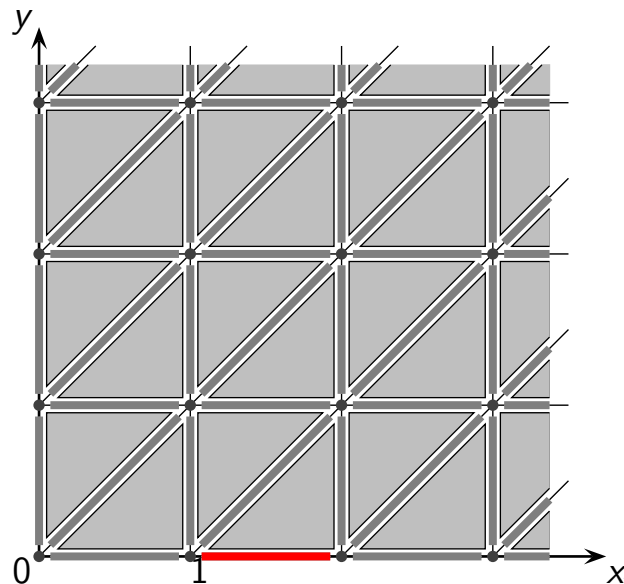
► Skip region graph



► Skip region graph

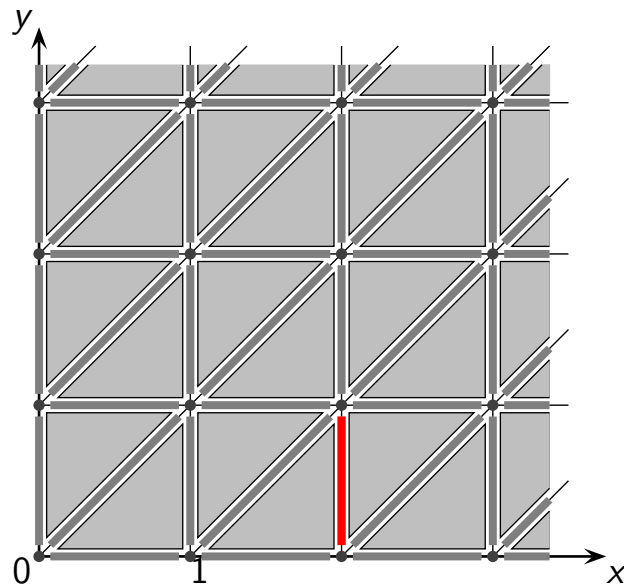


► Skip region graph

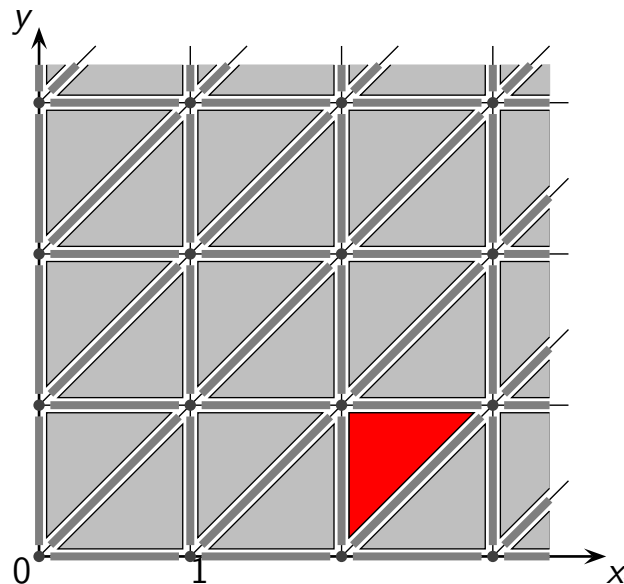


▶ Skip region graph

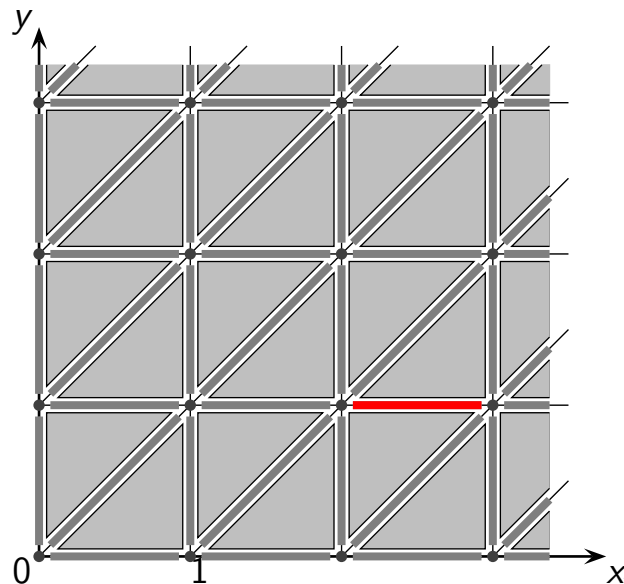
$y := 0$



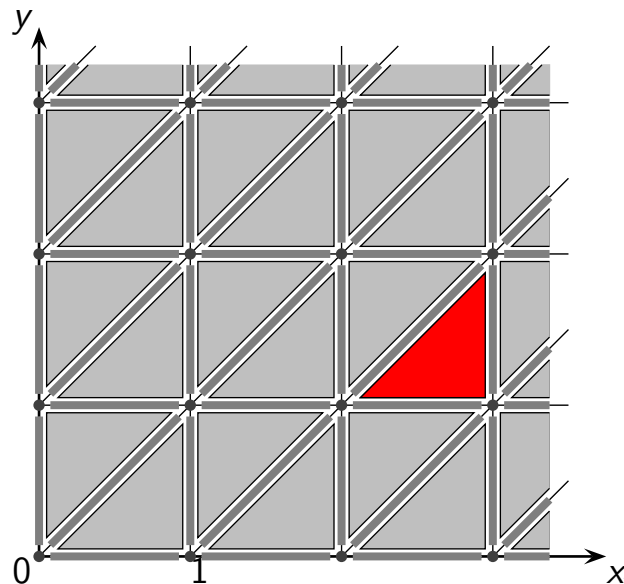
► Skip region graph



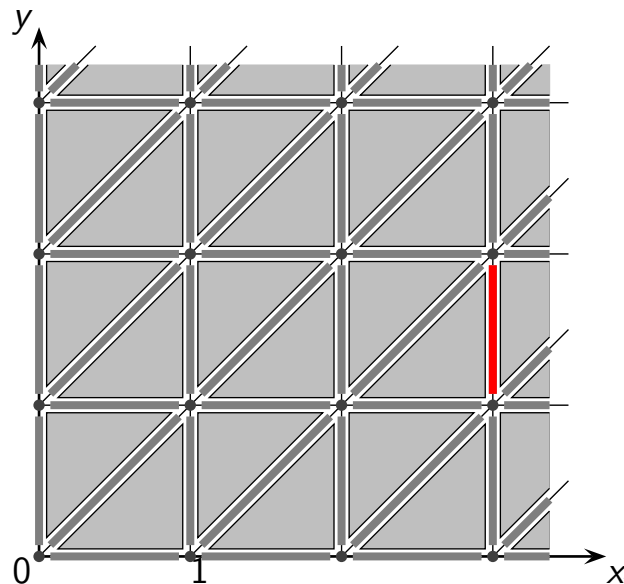
► Skip region graph



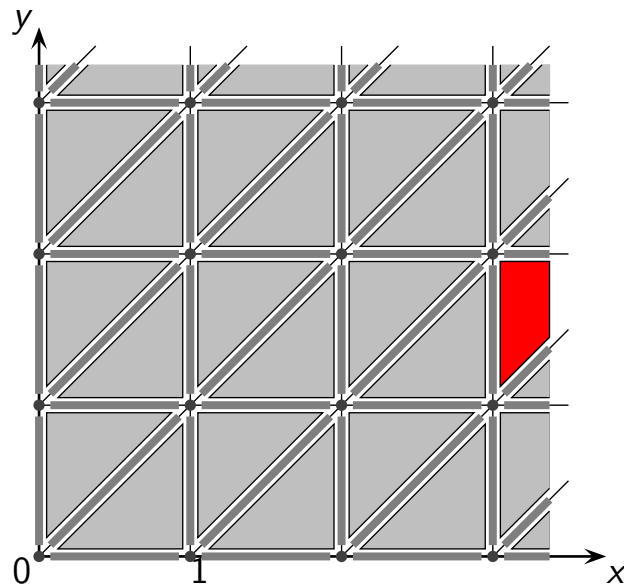
► Skip region graph



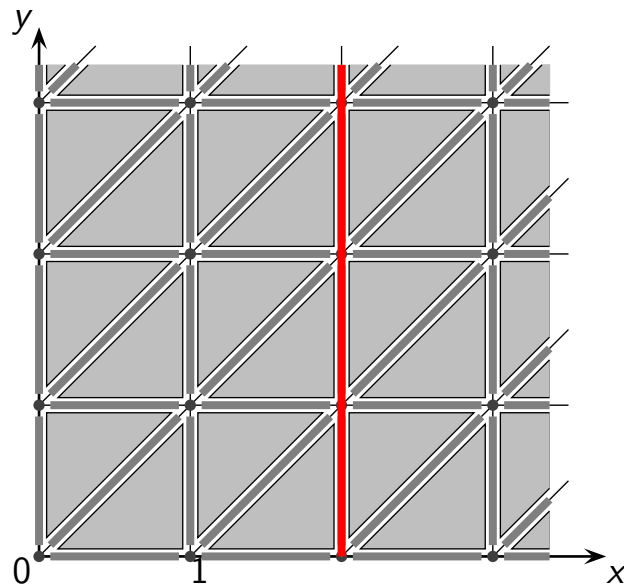
► Skip region graph



► Skip region graph

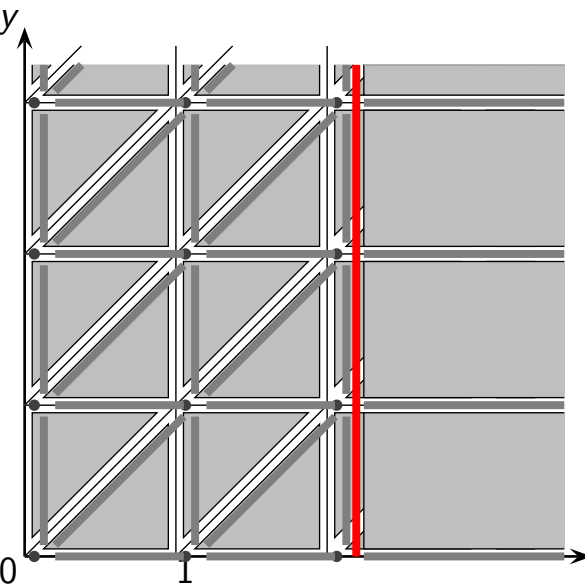


► Skip region graph



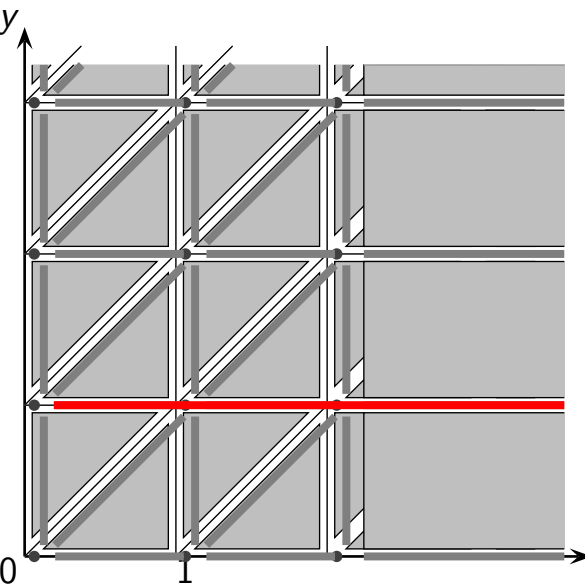
▶ Skip region graph

$$\max\{c \mid x \sim c\} = 2$$



► Skip region graph

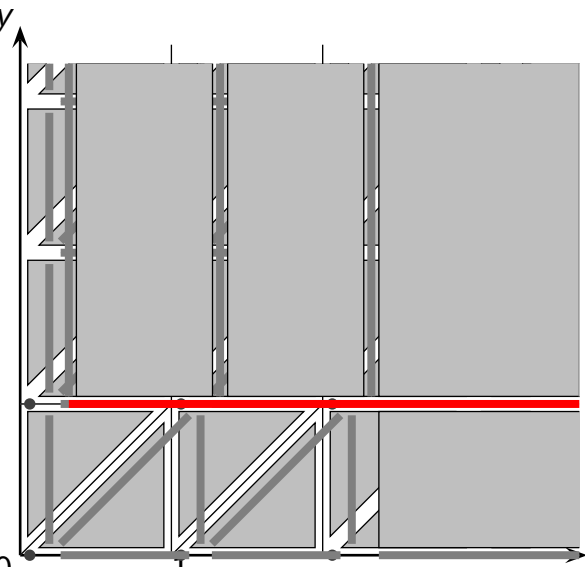
$$\max\{c \mid x \sim c\} = 2$$



► Skip region graph

$$\max\{c \mid x \sim c\} = 2$$

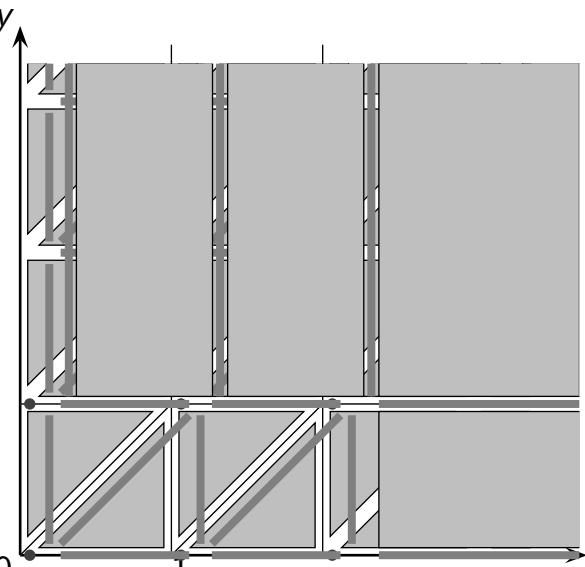
$$\max\{c \mid y \sim c\} = 1$$



► Skip region graph

$$\max\{c \mid x \sim c\} = 2$$

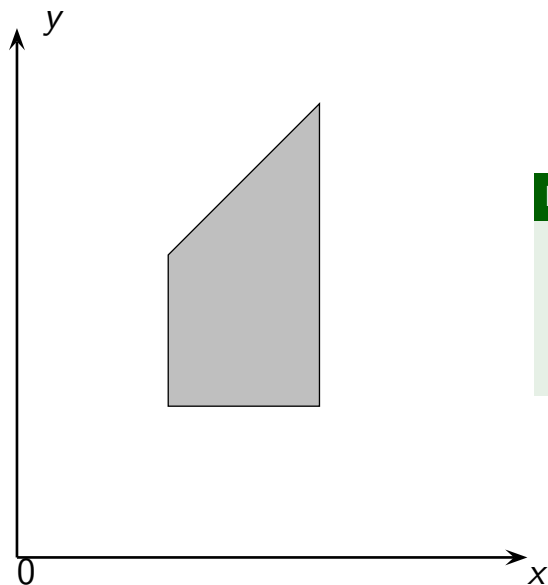
$$\max\{c \mid y \sim c\} = 1$$



► Skip region graph

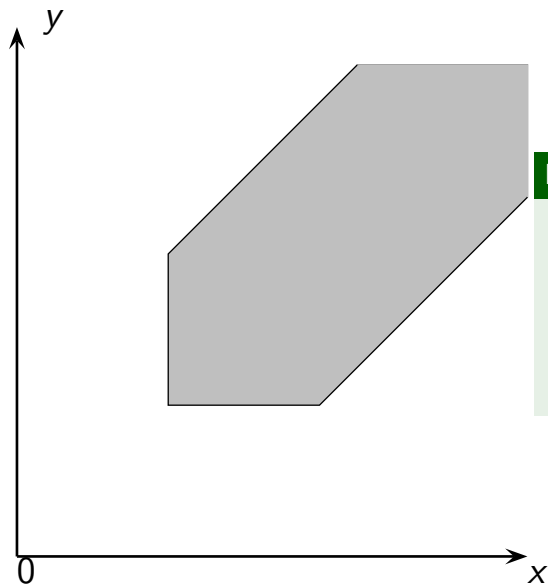
$$\max\{c \mid x \sim c\} = 2$$

$$\max\{c \mid y \sim c\} = 1$$



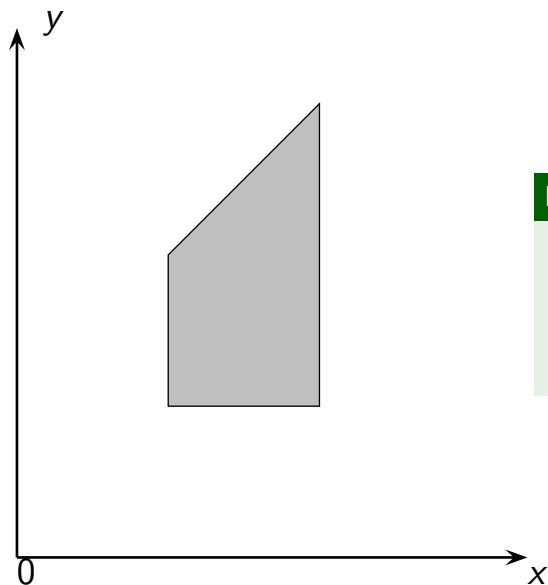
Example (Zone)

$$Z = \begin{cases} 1 \leq x \leq 2, \\ 1 \leq y \leq 3, \\ x - y \leq -1 \end{cases}$$



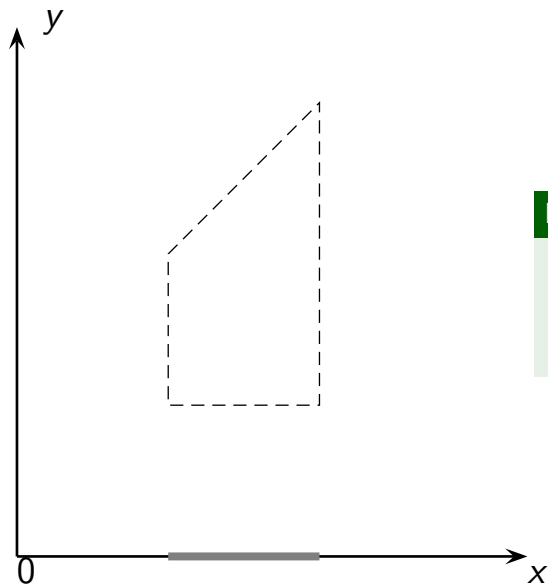
Example (Delay)

$$Z' = \begin{cases} 1 \leq x \leq \infty, \\ 1 \leq y \leq \infty, \\ 1 \leq x - y \leq -1 \end{cases}$$



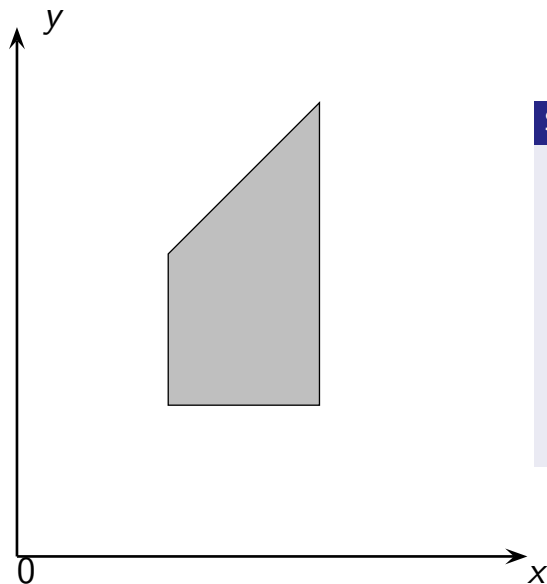
Example (Zone)

$$Z = \begin{cases} 1 \leq x \leq 2, \\ 1 \leq y \leq 3, \\ x - y \leq -1 \end{cases}$$



Example (Reset of y)

$$\{y\}Z = \begin{cases} 1 \leq x \leq 2, \\ y = 0 \end{cases}$$



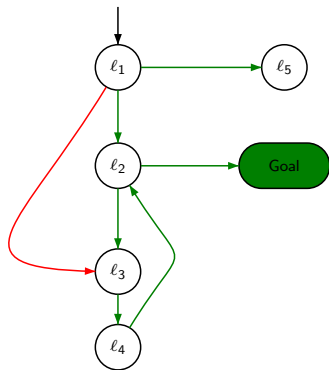
Simulation Graph

- $Z_0 = \{x = 0 \mid x \in X\} \xrightarrow{\text{inv}} \wedge \text{Inv}(l_0)$

- For each edge $e = (l, a, g, R, l')$,
 $(l, Z) \xrightarrow{a} (l', Z')$ with:

$$Z' = (\{R\}(Z \wedge g)) \xrightarrow{\text{inv}} \wedge \text{Inv}(l')$$

Plan



→ Uncontrollable (E_u)

→ Controllable (E_c)

Strategy:

$$F : \text{Runs}(A) \rightarrow E_c$$

Memoryless Strategy:

$$F : Q \rightarrow E_c$$

Winning Run:

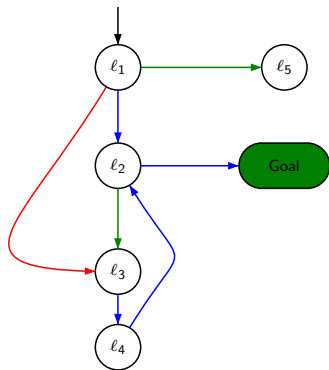
- Reachability:

$$\text{States}(\rho) \cap \text{Goal} \neq \emptyset$$

- Safety: $\text{States}(\rho) \cap \text{Bad} \neq \emptyset$

Winning Strategy:

$$\text{Runs}(A, F) \subseteq \text{WinRuns}$$



→ Uncontrollable (E_u)

→ Controllable (E_c)

Strategy:

$$F : \text{Runs}(A) \rightarrow E_c$$

Memoryless Strategy:

$$F : Q \rightarrow E_c$$

Winning Run:

- Reachability:

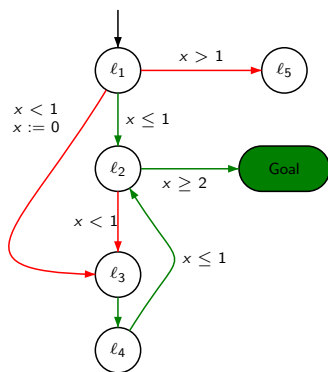
$$\text{States}(\rho) \cap \text{Goal} \neq \emptyset$$

- Safety: $\text{States}(\rho) \cap \text{Bad} \neq \emptyset$

Winning Strategy:

$$\text{Runs}(A, F) \subseteq \text{WinRuns}$$

Winning (memoryless) strategy



→ Uncontrollable (E_u)

→ Controllable (E_c)

Strategy:

$$F : \text{Runs}(A) \rightarrow E_c \cup \lambda$$

Memoryless Strategy:

$$F : Q \rightarrow E_c \cup \lambda$$

Winning Run:

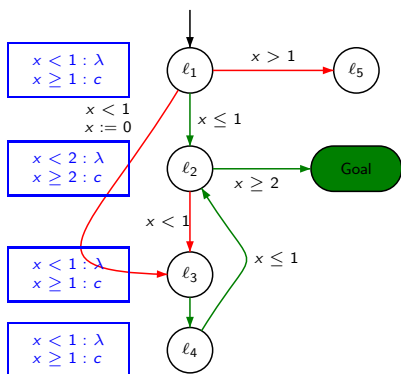
- Reachability:

$$\text{States}(\rho) \cap \text{Goal} \neq \emptyset$$

- Safety: $\text{States}(\rho) \cap \text{Bad} \neq \emptyset$

Winning Strategy:

$$\text{Runs}(A, F) \subseteq \text{WinRuns}$$



Strategy:

$$F : \text{Runs}(A) \rightarrow E_c \cup \lambda$$

Memoryless Strategy:

$$F : Q \rightarrow E_c \cup \lambda$$

Winning Run:

- Reachability: $\text{States}(\rho) \cap \text{Goal} \neq \emptyset$
- Safety: $\text{States}(\rho) \cap \text{Bad} \neq \emptyset$

Winning Strategy:

$$\text{Runs}(A, F) \subseteq \text{WinRuns}$$

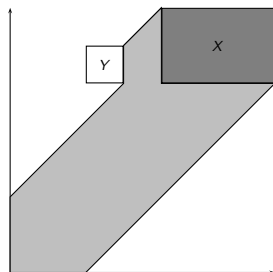
Winning (memoryless) strategy

$$\text{cPred}(X) = \{q \mid \exists q' \in X, q \rightarrow_c q'\}$$

$$\text{uPred}(X) = \{q \mid \exists q' \in X, q \rightarrow_u q'\}$$

$$\text{Pred}_t(X, Y) = \{q \mid \exists t, q^t \in X \text{ and } \forall s \leq t, q^s \in \bar{Y}\}$$

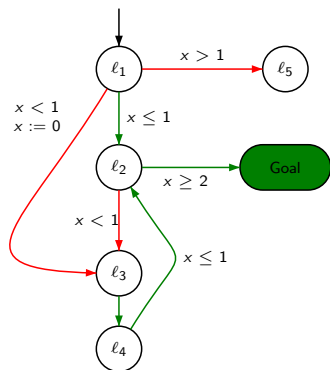
$$\pi(X) = \text{Pred}_t[X \cup \text{cPred}(X), \text{uPred}(\bar{X})]$$



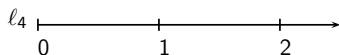
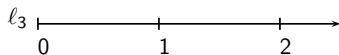
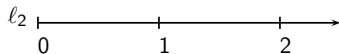
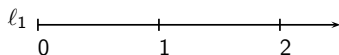
Theorem

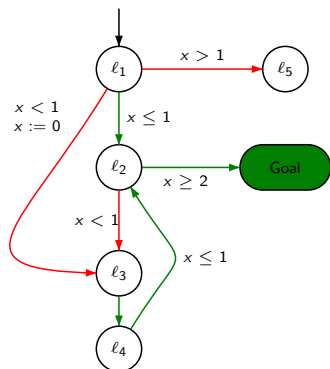
The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X) \cup \text{Goal}$$

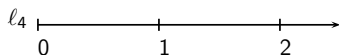
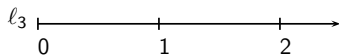
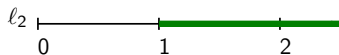
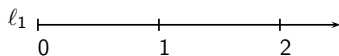


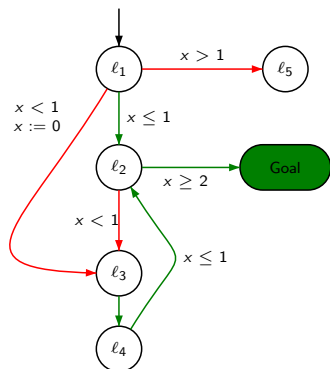
Backwards Fixed-point Computation



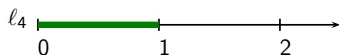
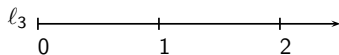
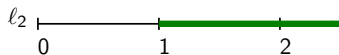
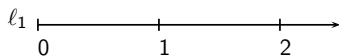


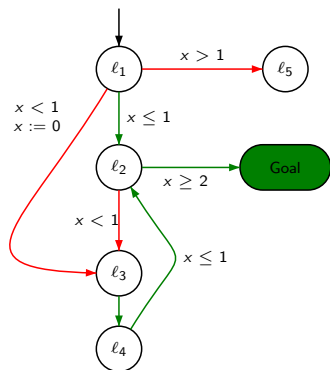
Backwards Fixed-point Computation



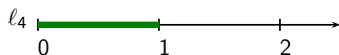
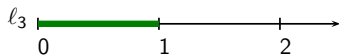
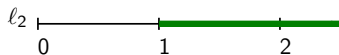
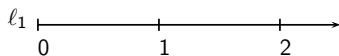


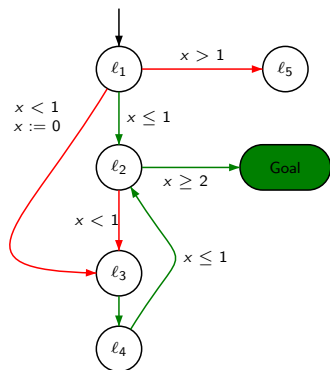
Backwards Fixed-point Computation



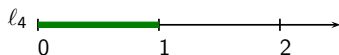
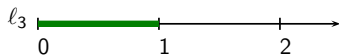
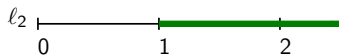
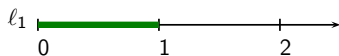


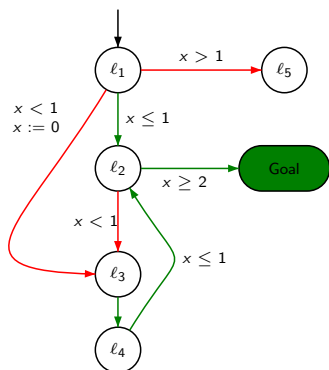
Backwards Fixed-point Computation





Backwards Fixed-point Computation

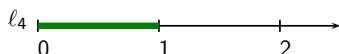
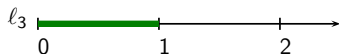
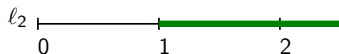
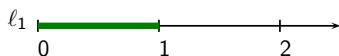


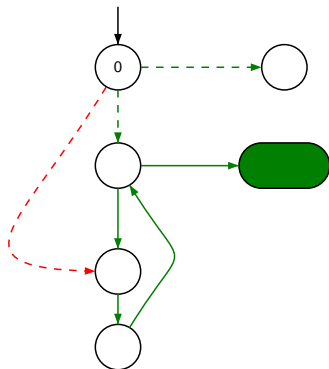


We want a **forward** and **on-the-fly** algorithm:

- Early termination
- Discrete variables

Backwards Fixed-point Computation





(Adapted from) Liu & Smolka, ICALP 1998

Initialization:

$Passed \leftarrow \{q_0\};$

$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$

$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$

$Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**

$e = (q, \alpha, q') \leftarrow pop(Waiting);$

if $q' \notin Passed$ **then**

$Passed \leftarrow Passed \cup \{q'\};$

$Depend[q'] \leftarrow \{(q, \alpha, q')\};$

$Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$

$Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$

if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$

else (* reevaluate *)

$Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$

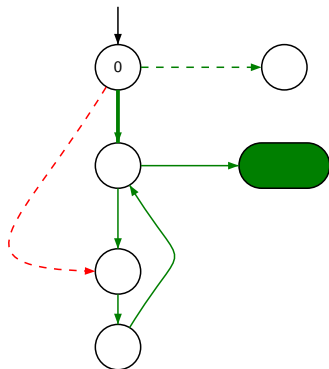
if Win^* **then**

$Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$

if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$

endif

► Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

Initialization:

$Passed \leftarrow \{q_0\};$

$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$

$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$

$Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**

$e = (q, \alpha, q') \leftarrow pop(Waiting);$

if $q' \notin Passed$ **then**

$Passed \leftarrow Passed \cup \{q'\};$

$Depend[q'] \leftarrow \{(q, \alpha, q')\};$

$Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$

$Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$

if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$

else (* reevaluate *)

$Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$

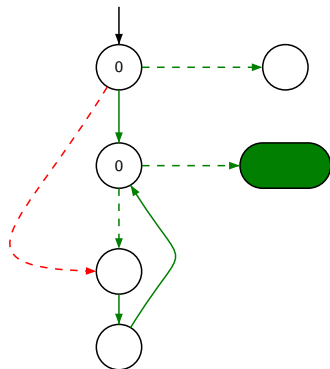
if Win^* **then**

$Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$

if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$

endif

» Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

» Skip algorithm

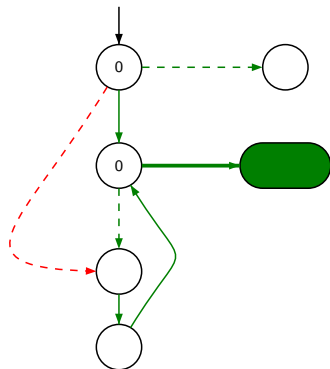
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

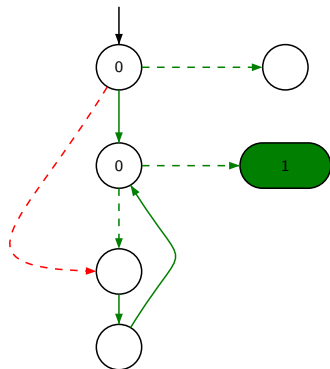
Initialization:

$Passed \leftarrow \{q_0\};$
 $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$
 $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$
 $Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else (* reevaluate *)
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif

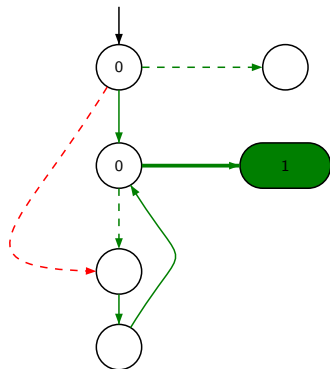
▶ Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

Initialization: $Passed \leftarrow \{q_0\};$ $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$ $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$ $Depend[q_0] \leftarrow \emptyset;$ **Main:****while** $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do** $e = (q, \alpha, q') \leftarrow pop(Waiting);$ **if** $q' \notin Passed$ **then** $Passed \leftarrow Passed \cup \{q'\};$ $Depend[q'] \leftarrow \{(q, \alpha, q')\};$ $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$ $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$ **if** $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$ **else** **(* reevaluate *)** $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$ **if** Win^* **then** $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$ **if** $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$ **endif**

» Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

Initialization:

$Passed \leftarrow \{q_0\};$

$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$

$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$

$Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**

$e = (q, \alpha, q') \leftarrow pop(Waiting);$

if $q' \notin Passed$ **then**

$Passed \leftarrow Passed \cup \{q'\};$

$Depend[q'] \leftarrow \{(q, \alpha, q')\};$

$Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$

$Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$

if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$

else **(* reevaluate *)**

$Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$

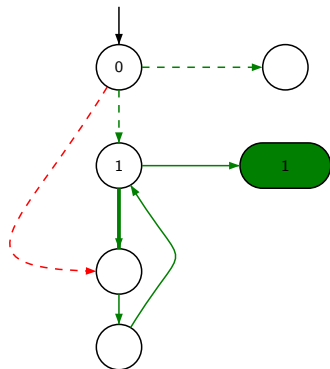
if Win^* **then**

$Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$

if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$

endif

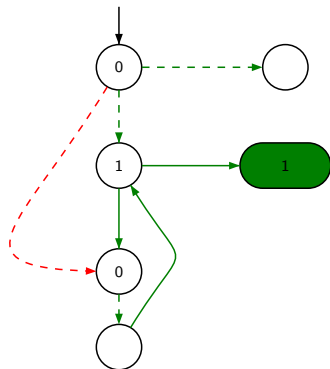
» Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

Initialization: $Passed \leftarrow \{q_0\};$ $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$ $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$ $Depend[q_0] \leftarrow \emptyset;$ **Main:****while** $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do** $e = (q, \alpha, q') \leftarrow pop(Waiting);$ **if** $q' \notin Passed$ **then** $Passed \leftarrow Passed \cup \{q'\};$ $Depend[q'] \leftarrow \{(q, \alpha, q')\};$ $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$ $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$ **if** $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$ **else** **(* reevaluate *)** $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$ **if** Win^* **then** $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$ **if** $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$ **endif**

» Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

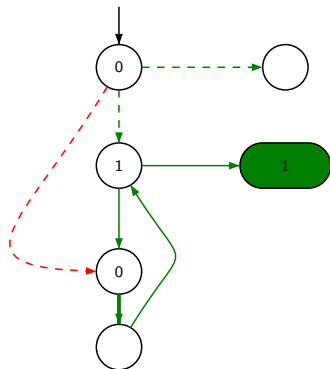
» Skip algorithm

Initialization:

$Passed \leftarrow \{q_0\};$
 $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$
 $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$
 $Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else (* reevaluate *)
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

» Skip algorithm

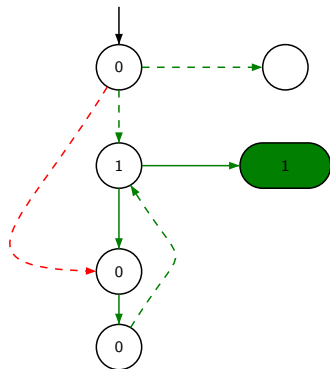
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

▶ Skip algorithm

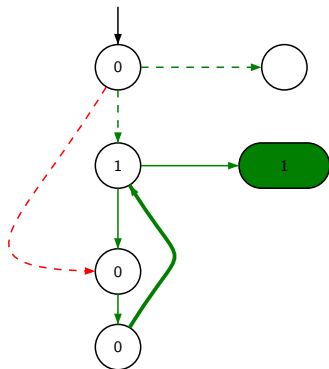
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

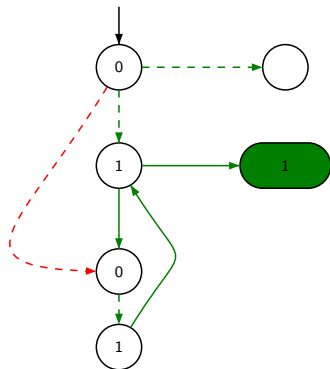
Initialization:

$Passed \leftarrow \{q_0\};$
 $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$
 $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$
 $Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else (* reevaluate *)
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif

» Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

▶ Skip algorithm

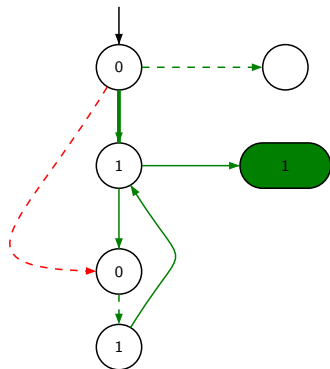
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

» Skip algorithm

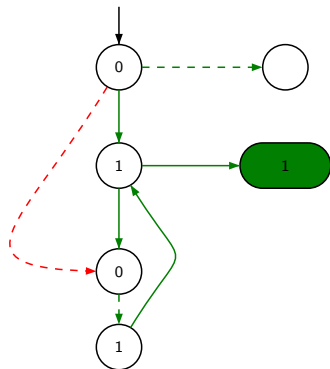
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

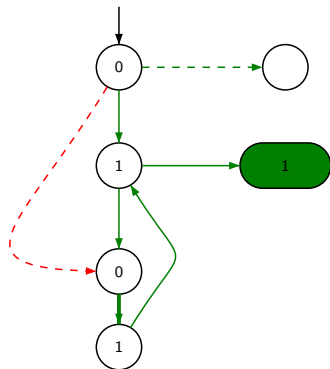
» Skip algorithm

Initialization:

$Passed \leftarrow \{q_0\};$
 $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$
 $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$
 $Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else (* reevaluate *)
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

» Skip algorithm

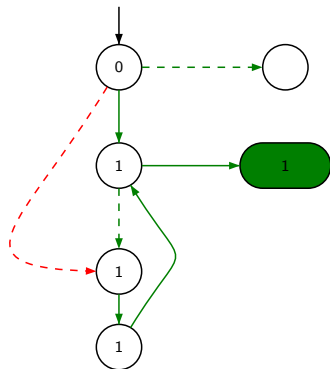
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

» Skip algorithm

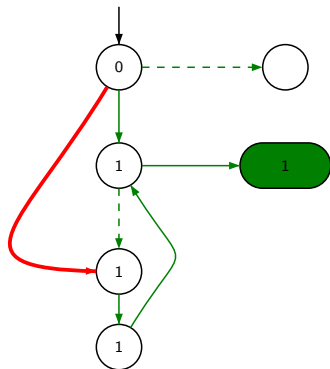
Initialization:

$$Passed \leftarrow \{q_0\};$$

$$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$$

$$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$$

$$Depend[q_0] \leftarrow \emptyset;$$
Main:
while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else **(* reevaluate *)**
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif



(Adapted from) Liu & Smolka, ICALP 1998

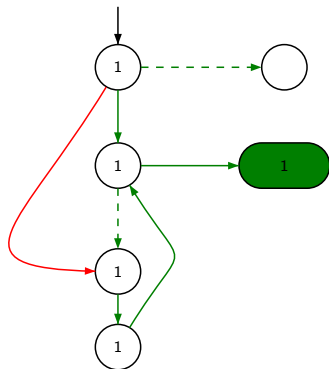
Initialization:

$Passed \leftarrow \{q_0\};$
 $Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$
 $Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$
 $Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**
 $e = (q, \alpha, q') \leftarrow pop(Waiting);$
if $q' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{q'\};$
 $Depend[q'] \leftarrow \{(q, \alpha, q')\};$
 $Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$
 $Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$
if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$
else (* reevaluate *)
 $Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$
if Win^* **then**
 $Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$
if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$
endif

▶ Skip algorithm



(Adapted from) Liu & Smolka, ICALP 1998

The algorithm is **linear**

Each edge is at most twice in the waiting list

▶ Skip algorithm

Initialization:

$Passed \leftarrow \{q_0\};$

$Waiting \leftarrow \{(q_0, \alpha, q') \mid \alpha \in Act \ q \xrightarrow{\alpha} q'\};$

$Win[q_0] \leftarrow (q_0 \in Goal ? 1 : 0);$

$Depend[q_0] \leftarrow \emptyset;$

Main:

while $((Waiting \neq \emptyset) \wedge Win[q_0] \neq 1)$ **do**

$e = (q, \alpha, q') \leftarrow pop(Waiting);$

if $q' \notin Passed$ **then**

$Passed \leftarrow Passed \cup \{q'\};$

$Depend[q'] \leftarrow \{(q, \alpha, q')\};$

$Win[q'] \leftarrow (q' \in Goal ? 1 : 0);$

$Waiting \leftarrow Waiting \cup \{(q', \alpha, q'') \mid q' \xrightarrow{\alpha} q''\};$

if $Win[q']$ **then** $Waiting \leftarrow Waiting \cup \{e\};$

else (* reevaluate *)

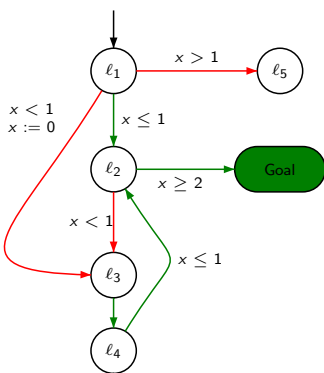
$Win^* \leftarrow \bigwedge_{q \xrightarrow{u} u} Win[u] \wedge \bigvee_{q \xrightarrow{c} w} Win[w];$

if Win^* **then**

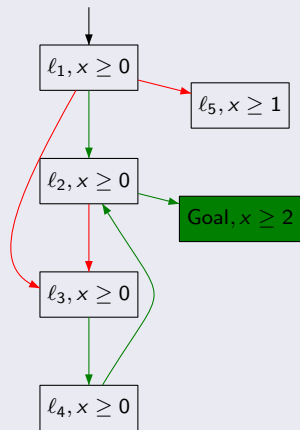
$Waiting \leftarrow Waiting \cup Depend[q]; Win[q] \leftarrow 1;$

if $Win[q'] = 0$ **then** $Depend[q'] \leftarrow Depend[q'] \cup \{e\};$

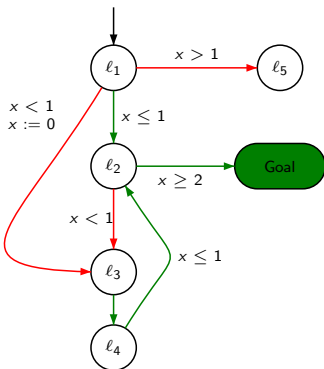
endif



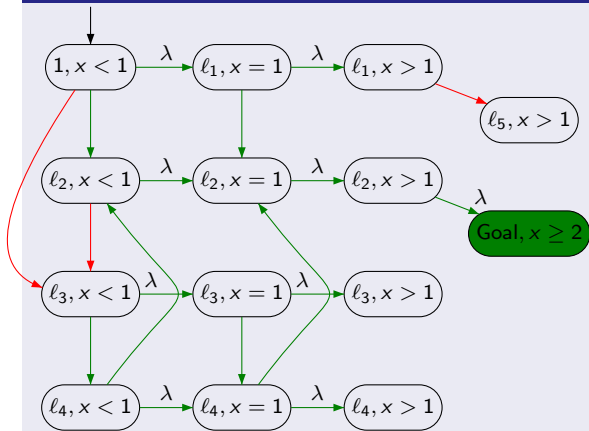
First attempt



Simulation graph

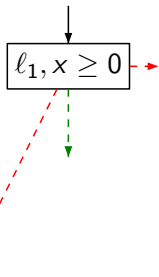


Second attempt



S. TRIPAKIS FME 1999, RTTOOLS 2001: apply an untimed on-the-fly algorithm on a stable partitioning

Plan



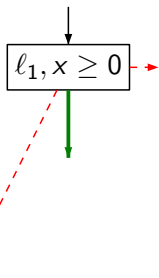
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



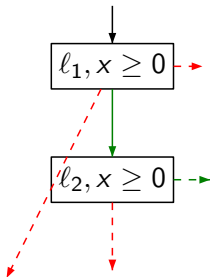
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \bar{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



▶ Skip algorithm

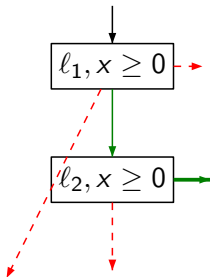
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s0 ∉ Win[S0])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ≥0X);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Postα(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  end (* reevaluate *)
  Win* ← Predt(Win[S] ∪ ⋃Sc→T Predc(Win[T]),
              ⋃Su→T Predu(T \ Win[T])) ∩ S;
  if (Win[S] ⊂ Win*) then
    Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
    Depend[S'] ← Depend[S'] ∪ {e};
  endif
endwhile
  
```



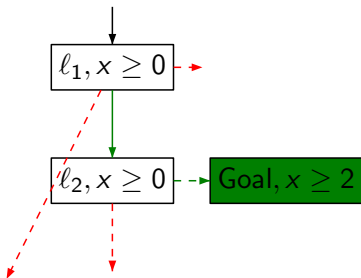
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \bar{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



▶ Skip algorithm

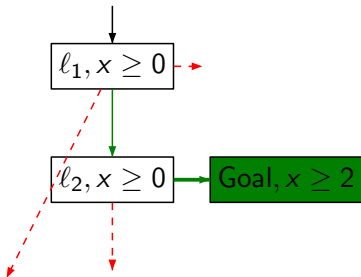
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ≥0x);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Postα(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Predt(Win[S] ∪ ⋃Sc→T Predc(Win[T]),
                ⋃Su→T Predu(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endwhile
  endwhile
  
```



▶ Skip algorithm

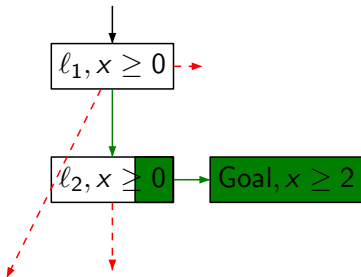
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s0 ∉ Win[S0])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ≥0x);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Postα(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Predt(Win[S] ∪ ⋃Sc→T Predc(Win[T]),
                ⋃Su→T Predu(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



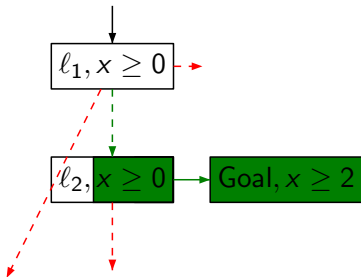
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \bar{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



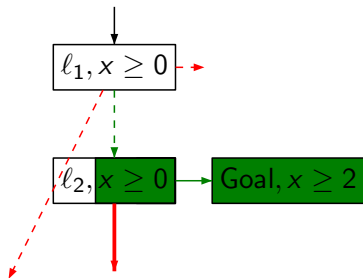
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \bar{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_c(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



▶ Skip algorithm

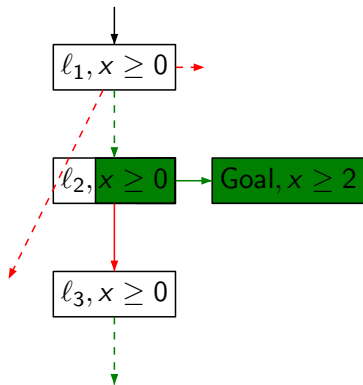
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^X);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S' →_T} Pred_c(Win[T]),
                  ⋃_{S' →_T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



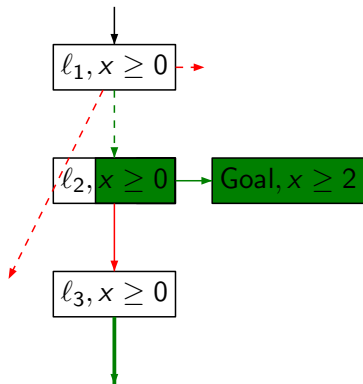
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



▶ Skip algorithm

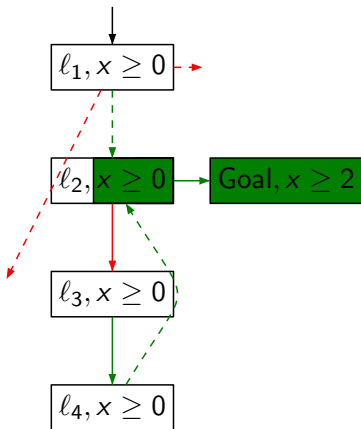
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^X);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')'};
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S' →_T} Pred_c(Win[T]),
                  ⋃_{S' →_T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



▶ Skip algorithm

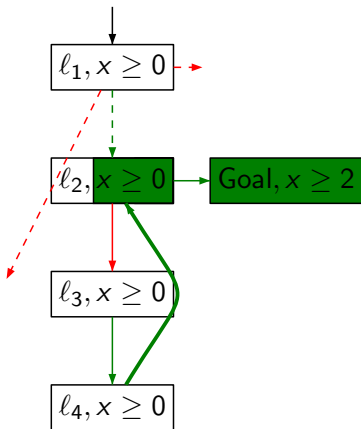
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^x);
    Waiting ← Waiting ∪ {(S', α, S'') ∣ S'' = Post_α(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S →_c T} Pred_c(Win[T]),
                  ⋃_{S →_u T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



► Skip algorithm

Initialization:

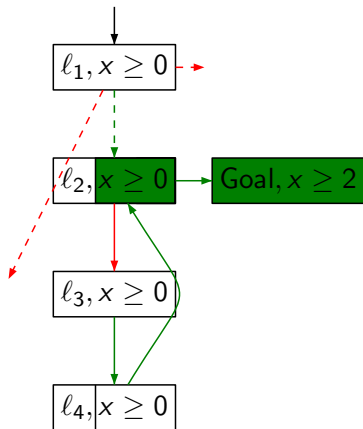
$$\begin{aligned} \text{Passed} &\leftarrow \{S_0\} \text{ where } S_0 = \{(\ell_0, \vec{0})\}^{\nearrow}; \\ \text{Waiting} &\leftarrow \{(S_0, \alpha, S') \mid S' = \text{Post}_\alpha(S_0)^{\nearrow}\}; \\ \text{Win}[S_0] &\leftarrow S_0 \cap (\{\text{Goal}\} \times \mathbb{R}_{\geq 0}^x); \\ \text{Depend}[S_0] &\leftarrow \emptyset; \end{aligned}$$

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^x);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')^{\nearrow}};
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S' →_c T} Pred_c(Win[T]),
                  ⋃_{S' →_u T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile

```



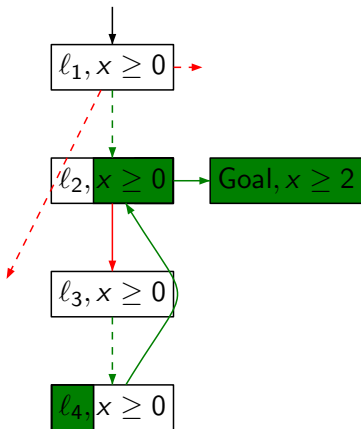
► Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S]) \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T])$;
 $\quad \cup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T]) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



► Skip algorithm

Initialization:

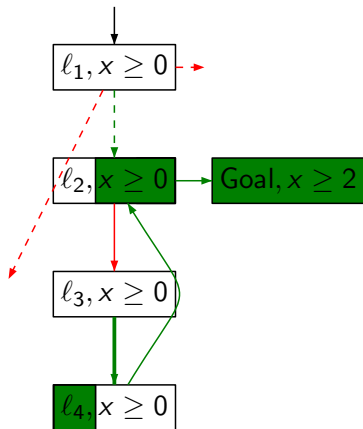
$$\begin{aligned} Passed &\leftarrow \{S_0\} \text{ where } S_0 = \{(\ell_0, \vec{0})\}^{\nearrow}; \\ Waiting &\leftarrow \{(S_0, \alpha, S') \mid S' = \text{Post}_\alpha(S_0)^{\nearrow}\}; \\ Win[S_0] &\leftarrow S_0 \cap (\{\text{Goal}\} \times \mathbb{R}_{\geq 0}^x); \\ Depend[S_0] &\leftarrow \emptyset; \end{aligned}$$

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^x);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')^{\nearrow}};
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_c(Win[S] ∪ ⋃_{S' →_c T} Pred_c(Win[T]),
                  ⋃_{S' →_u T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile

```



► Skip algorithm

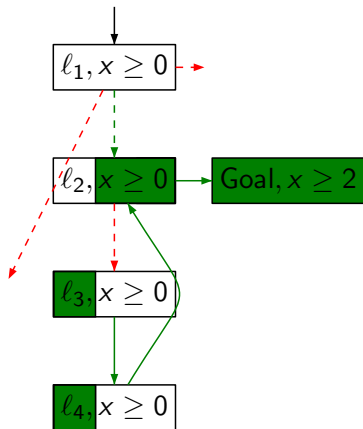
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s0 ∉ Win[S0])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ≥0x);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Postα(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Predt(Win[S] ∪ ⋃Sc→T Predc(Win[T]),
                ⋃Su→T Predu(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



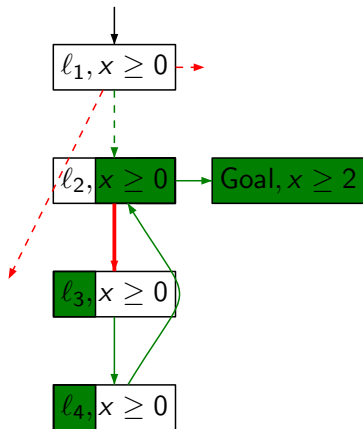
► Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = \text{Post}_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{\text{Goal}\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{\text{Goal}\} \times \mathbb{R}_{\geq 0}^x)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = \text{Post}_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow \text{Pred}_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} \text{Pred}_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} \text{Pred}_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



▶ Skip algorithm

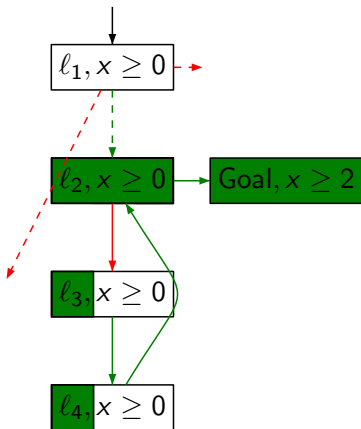
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ≥0x);
    Waiting ← Waiting ∪ {(S', α, S'') ∣ S'' = Postα(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Predt(Win[S] ∪ ⋃Sc→T Predc(Win[T]),
                ⋃Su→T Predu(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



► Skip algorithm

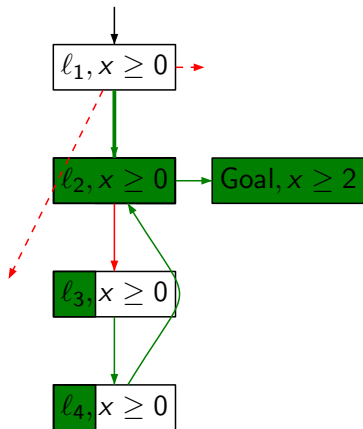
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^x);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S' →_c T} Pred_c(Win[T]),
                  ⋃_{S' →_u T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



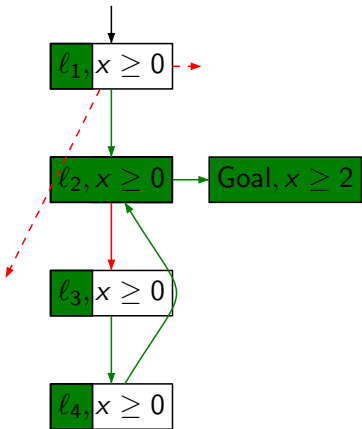
► Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S]) \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T])$;
 $\quad \quad \quad \bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T]) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



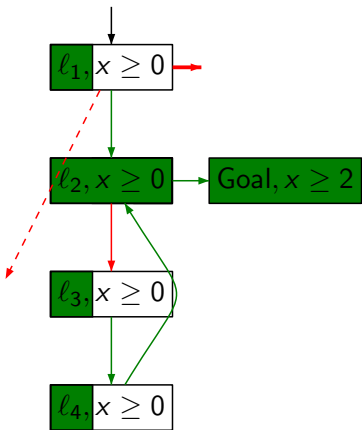
▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S]) \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T])$;
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T]) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



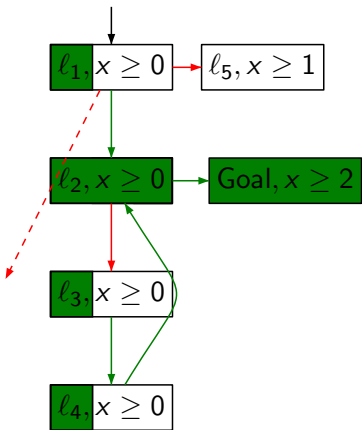
► Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^x)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T]),$
 $\bigcup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T])) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



► Skip algorithm

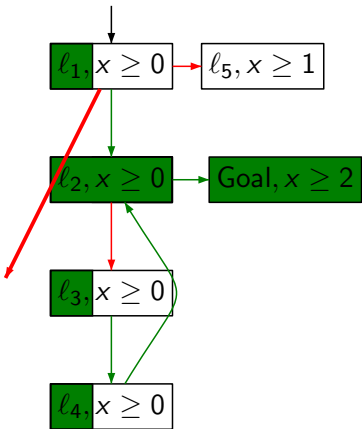
Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^X);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')'};
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S' →_c T} Pred_c(Win[T]),
                  ⋃_{S' →_u T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endif
endwhile
  
```



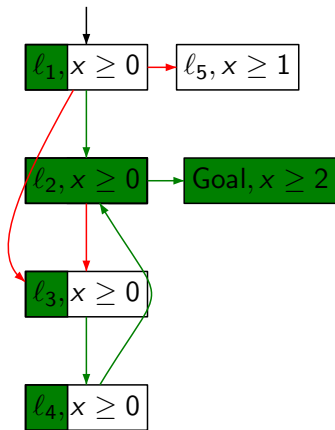
► Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
 $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
if $S' \notin Passed$ **then**
 $Passed \leftarrow Passed \cup \{S'\}$;
 $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
 $Win[S'] \leftarrow S' \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = Post_\alpha(S')'\}$;
if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
else (* reevaluate *)
 $Win^* \leftarrow Pred_t(Win[S]) \cup \bigcup_{S \xrightarrow{c} T} Pred_c(Win[T])$;
 $\quad \cup_{S \xrightarrow{u} T} Pred_u(T \setminus Win[T]) \cap S$;
if $(Win[S] \subsetneq Win^*)$ **then**
 $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
 $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
endif
endwhile



▶ Skip algorithm

Initialization:

$Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \vec{0})\}'$;
 $Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = Post_\alpha(S_0)'\}$;
 $Win[S_0] \leftarrow S_0 \cap (\{Goal\} \times \mathbb{R}_{\geq 0}^X)$;
 $Depend[S_0] \leftarrow \emptyset$;

Main:

```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ_{≥0}^X);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Post_α(S')' };
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  else (* reevaluate *)
    Win* ← Pred_t(Win[S] ∪ ⋃_{S' →_c T} Pred_c(Win[T]),
                  ⋃_{S' →_u T} Pred_u(T \ Win[T])) ∩ S;
    if (Win[S] ⊂ Win*) then
      Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
      Depend[S'] ← Depend[S'] ∪ {e};
    endif
  endwhile
  
```

Theorem

The following distribution law holds:

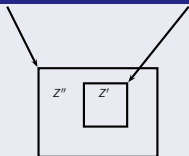
$$\text{Pred}_t(\bigcup_i G_i, \bigcup_j B_j) = \bigcup_i \bigcap_j \text{Pred}_t(G_i, B_j) \quad (1)$$

Theorem

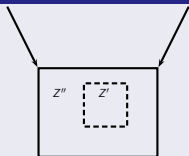
If B is a convex set, then:

$$\text{Pred}_t(G, B) = (G \setminus B) \cup ((G \cap B) \setminus B) \quad (2)$$

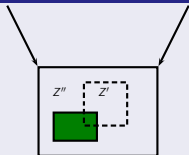
Inclusion checking



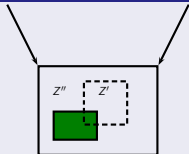
Inclusion checking



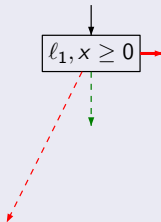
Inclusion checking



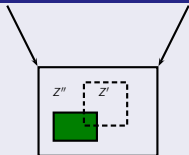
Inclusion checking



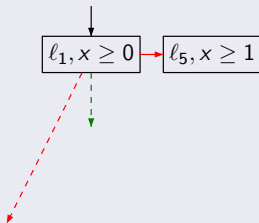
Losing states



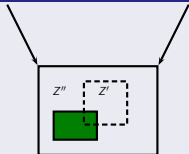
Inclusion checking



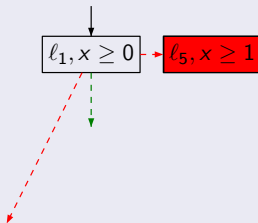
Losing states



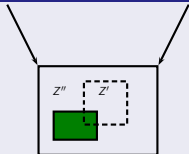
Inclusion checking



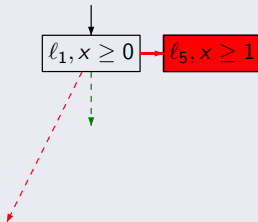
Losing states



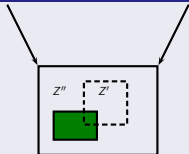
Inclusion checking



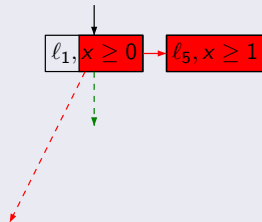
Losing states



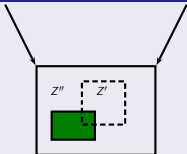
Inclusion checking



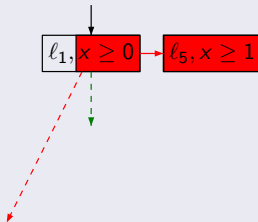
Losing states



Inclusion checking



Losing states



Pruning

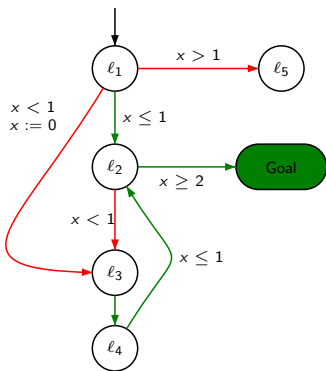
Main:

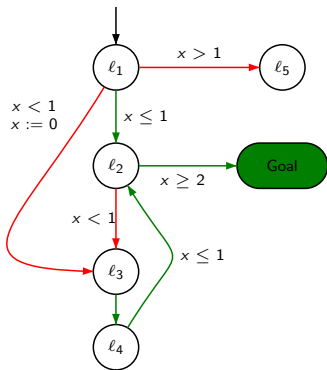
```

while ((Waiting ≠ ∅) ∧ (s₀ ∉ Win[S₀])) do
  e = (S, α, S') ← pop(Waiting);
  if Win[S] ⊆ S then
    if S' ∉ Passed then
      (...)
    else (* reevaluate *)
      (...)
    endif
  endif
endwhile

```

Plan



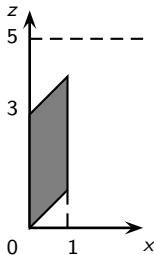


Assumptions:

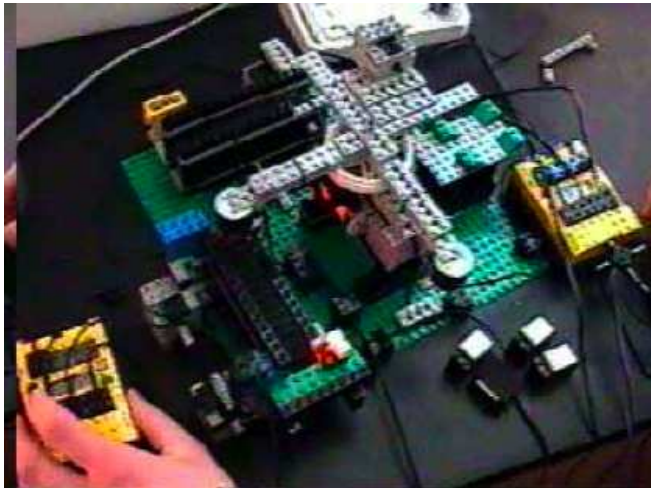
- The game is winning
- We know an upper bound B of the minimum time needed to reach the goal

Modifications:

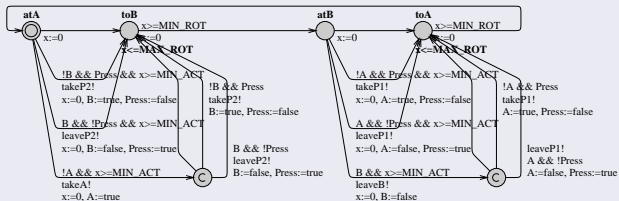
- Add a clock z (unconstrained at the beginning)
- Add a global invariant $z \leq B$



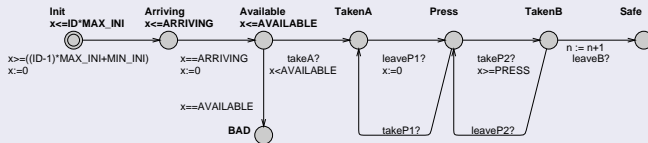
Plan



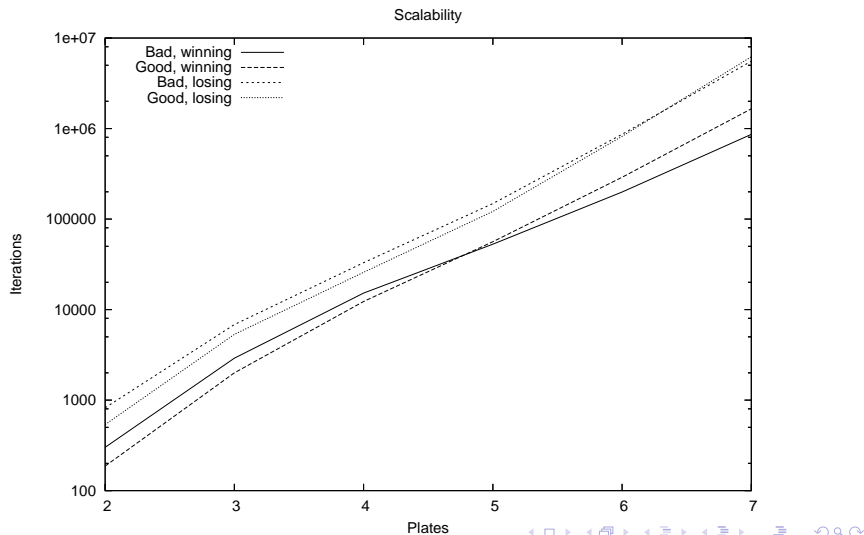
Model of the robot



Model of the plates



Plates		Basic		Basic +inc		Basic +inc +pruning		Basic+lose +inc +pruning		Basic+lose +inc +topt	
		time	mem	time	mem	time	mem	time	mem	time	mem
2	win	0.0s	1M	0.0s	1M	0.0s	1M	0.0s	1M	0.04s	1M
	lose	0.0s	1M	0.0s	1M	0.0s	1M	0.0s	1M	n/a	n/a
3	win	0.5s	19M	0.0s	1M	0.0s	1M	0.1s	1M	0.27s	4M
	lose	1.1s	45M	0.1s	1M	0.0s	1M	0.2s	3M	n/a	n/a
4	win	33.9s	1395M	0.2s	8M	0.1s	6M	0.4s	5M	1.88s	13M
	lose	-	-	0.5s	11M	0.4s	10M	0.9s	9M	n/a	n/a
5	win	-	-	3.0s	31M	1.5s	22M	2.0s	16M	13.35s	59M
	lose	-	-	11.1s	61M	5.9s	46M	7.0s	41M	n/a	n/a
6	win	-	-	89.1s	179M	38.9s	121M	12.0s	63M	220.3s	369M
	lose	-	-	699s	480M	317s	346M	135.1s	273M	n/a	n/a
7	win	-	-	3256s	1183M	1181s	786M	124s	319M	6188s	2457M
	lose	-	-	-	-	16791s	2981M	4075s	2090M	n/a	n/a



Conclusion and Future work

Conclusions:

- Successful development of a truly on-the-fly algorithm for reachability and safety games
- efficient implementation using the UPPAAL DBM library
- Algorithm suitable for other winning criteria

Future work:

- Integration with the UPPAAL GUI.
- Guiding of the exploration by ordering the waiting list (heuristics)
- Distributed implementation
- Partial observability