

A New and Simpler Leader Election Protocol (IEEE 1394)

J.R. Abrial, D. Cansell, D. Méry

May 10, 2006

This Session

- Background :-)
- An informal presentation of the protocol :-)
- Step by step formal design :-|
- Replaying with models
- Short Conclusion. :-)

A methodology to develop distributed algorithms

- Establishing the **mathematical framework**

A methodology to develop distributed algorithms

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**

A methodology to develop distributed algorithms

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**
- Resolving the same problem on a **step by step basis**
by introducing the essence of the algorithm

A methodology to develop distributed algorithms

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**
- Resolving the same problem on a **step by step basis** by introducing the essence of the algorithm
- Involving communication **by means of messages**

A methodology to develop distributed algorithms

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**
- Resolving the same problem on a **step by step basis** by introducing the essence of the algorithm
- Involving communication **by means of messages**
- Towards the **localization of data structures**

Replaying with a development

Most of cases we **restart** a modelisation from **scratsh** to obtain a **well-known** algorithm!

- Can we **reuse** or **improve** a modelisation ?
- Can we modify models to obtain **new** algorithm ? **sim-pler** one ?
- Can we use a B tool to **improve** new idea ?

Replaying with a development

Most of cases we **restart** a modelisation from **scratsh** to obtain a **well-known** algorithm!

- Can we **reuse** or **improve** a modelisation ?
- Can we modify models to obtain **new** algorithm ? **sim-pler** one ?
- Can we use a B tool to **improve** new idea ?

The answer is **YES**

IEEE 1394 High Performance Serial Bus (FireWire)

- It is an **international standard**
- There exists a **widespread commercial interest** in its correctness
- Sun, Apple, Philips, Microsoft, Sony, etc **involved in its development**
- Made of **three layers** (physical, link, transaction)
- The protocol under study is the **Tree Identify Protocol**
- Situated in the **Bus Reset phase** of the physical layer

The Problem (1)

- The bus is used to transport digitized **video and audio signals**
- It is **“hot-pluggable”**
- Devices and peripherals can be **added and removed at any time**
- Such changes are followed by a **bus reset**
- The **leader election** takes place after a bus reset in the network
- A leader needs to be chosen to act as the **manager of the bus**

The Problem (2)

- After a bus reset: all nodes in the network have **equal status**
- A node **only knows** to which nodes it is **directly connected**
- The network is **connected**
- The network is **acyclic**

References (1)

BASIC

- IEEE. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995.* 1995
- IEEE. *IEEE Standard for a High Performance Serial Bus (supplement). Std 1394a-2000.* 2000

References (2)

GENERAL

- N. Lynch. *Distributed Algorithms*. Morgan Kaufmann. 1996
- R. G. Gallager et al. *A Distributed Algorithm for Minimum Weight Spanning Trees*. IEEE Trans. on Prog. Lang. and Systems. 1983.

References (3)

MODEL CHECKING

- D.P.L. Simons et al. *Mechanical Verification of the IEE 1394a Root Contention Protocol using Uppaal2* Springer International Journal of Software Tools for Technology Transfer. 2001
- H. Toetenel et al. *Parametric verification of the IEEE 1394a Root Contention Protocol using LPMC* Proceedings of the 7th International Conference on Real-time Computing Systems and Applications. IEEE Computer Society Press. 2000

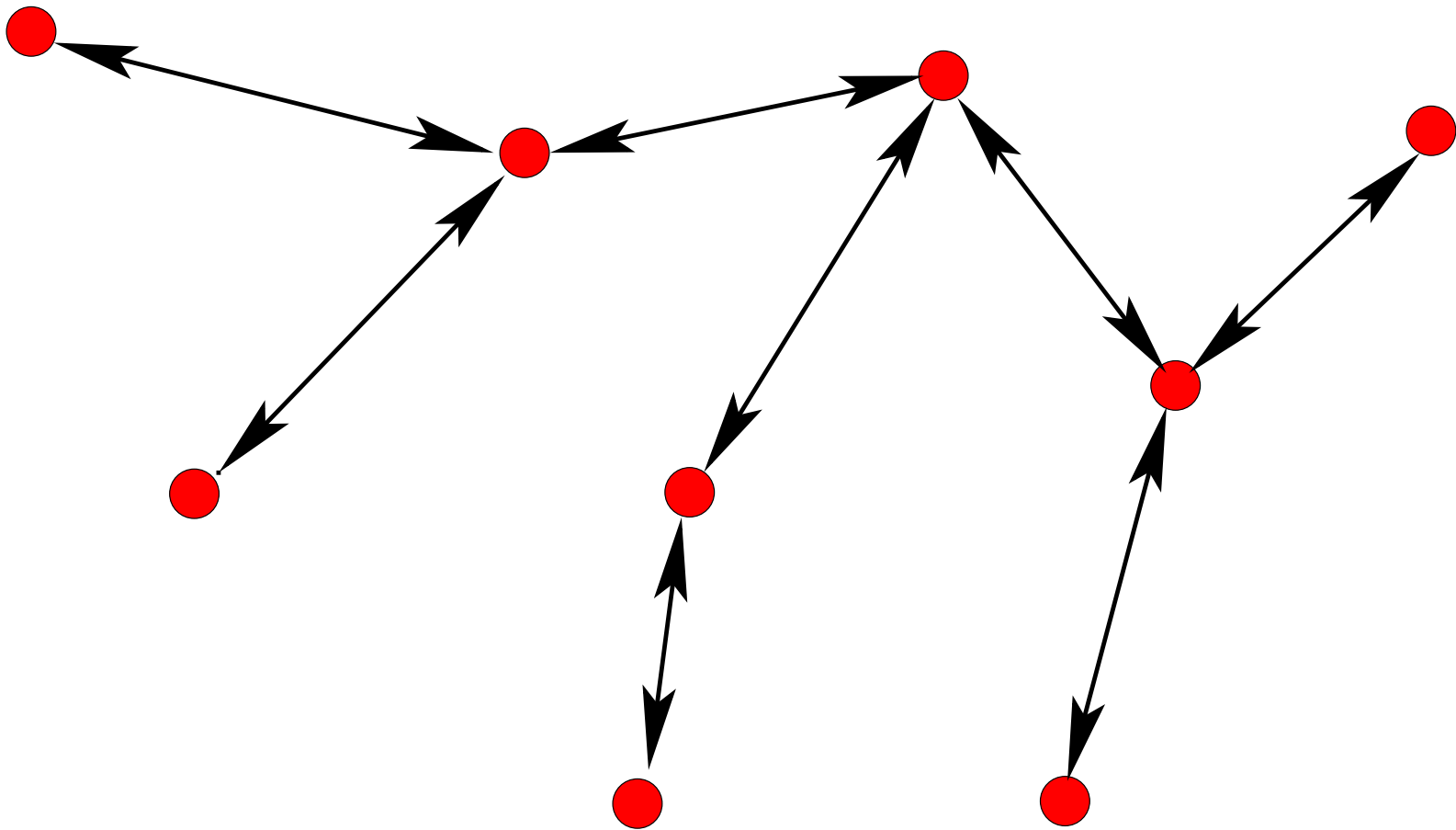
References (4)

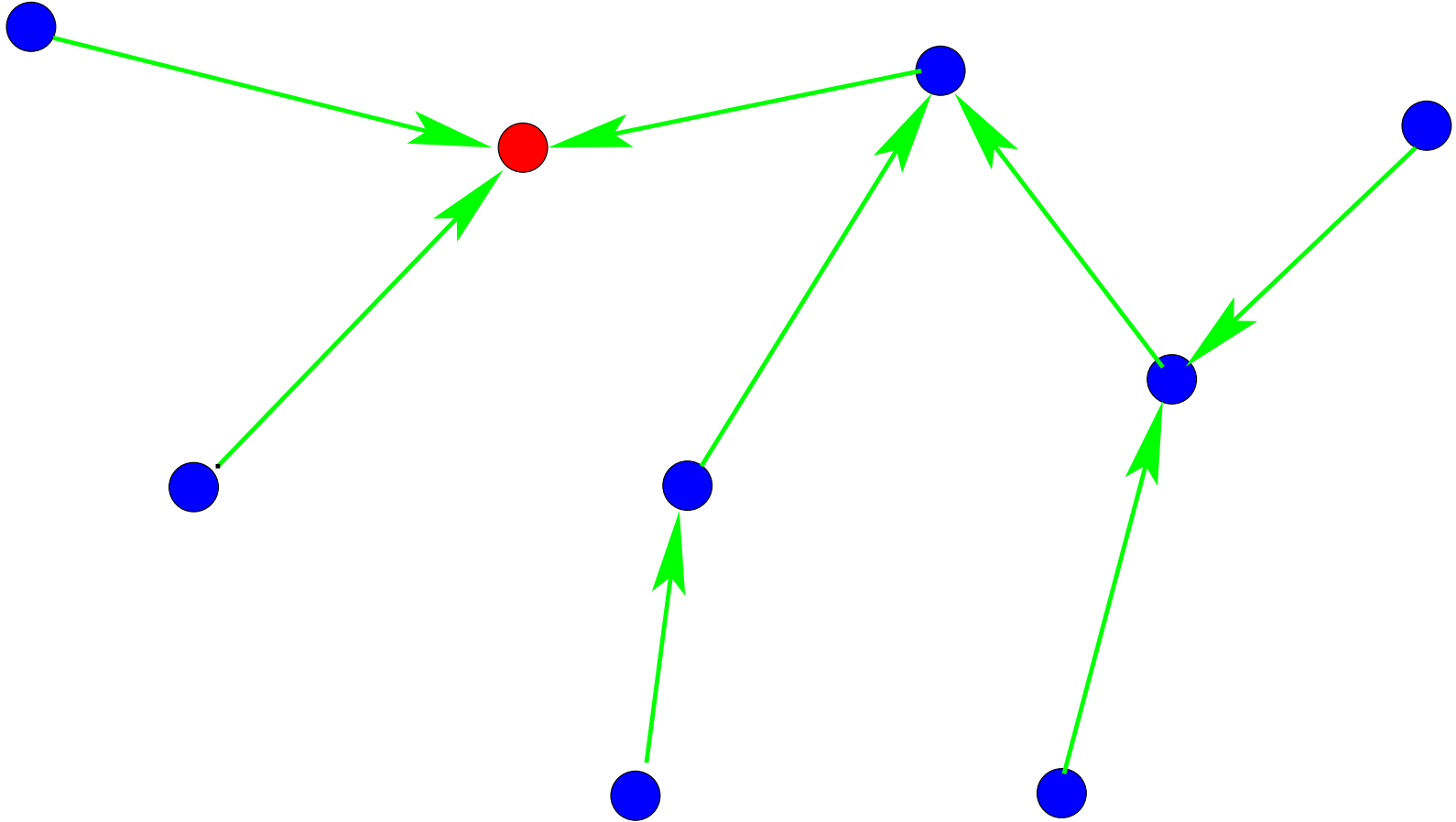
THEOREM PROVING

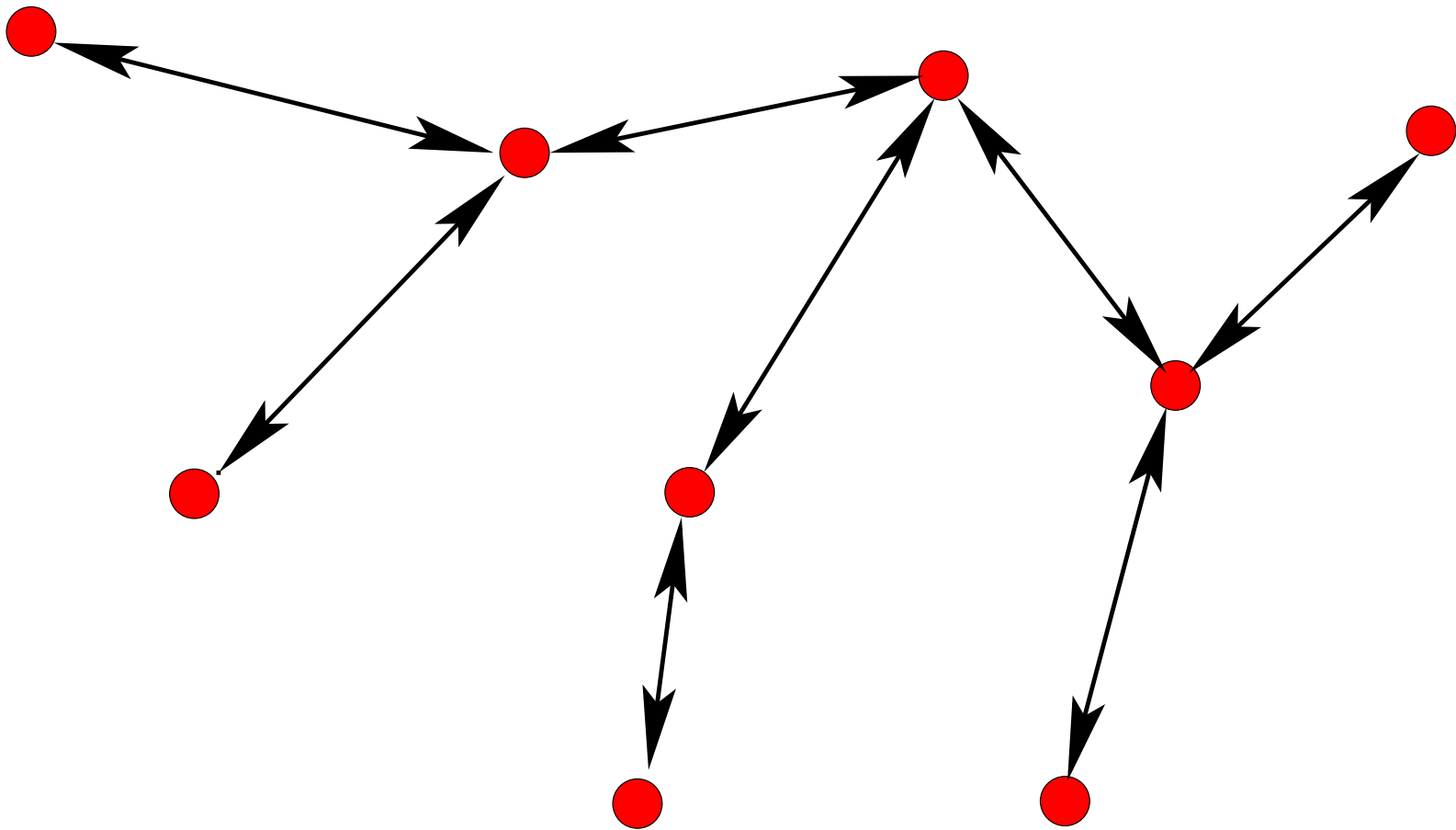
- M. Devillers et al. *Verification of the Leader Election: Formal Method Applied to IEEE 1394*. Formal Methods in System Design. 2000
- J.R. Abrial et al. *A Mechanically Proved and Incremental Development of IEEE 1394*. To be published 2002

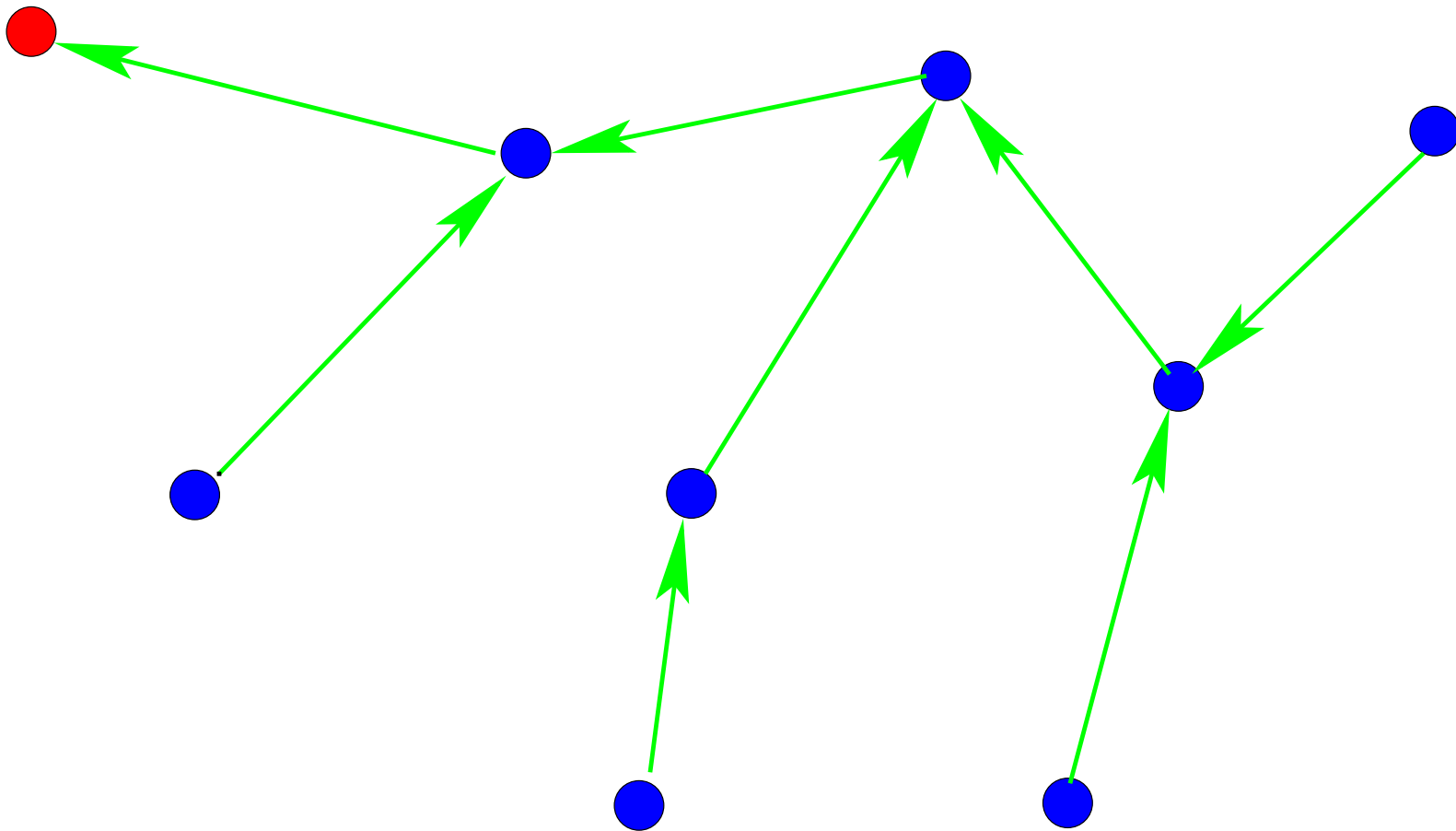
Informal Abstract Properties of the Protocol

- We are given a **connected** and **acyclic** network of nodes
- Nodes are linked by **bidirectional channels**
- We want to have one node being elected **the leader** in a finite time
- This is to be done in a **distributed** and **non-deterministic** way
- Next are two distinct **abstract animations** of the protocol



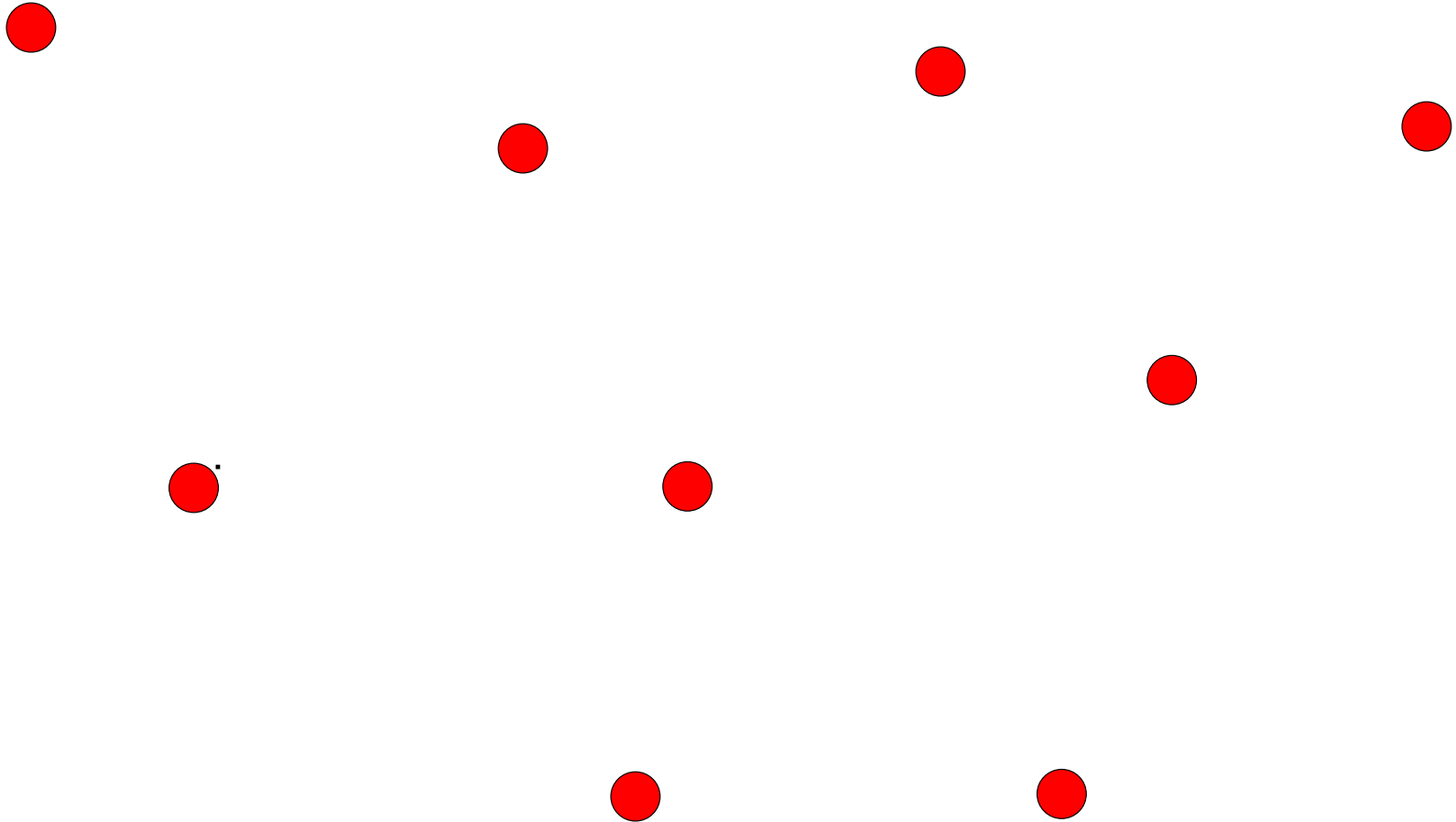




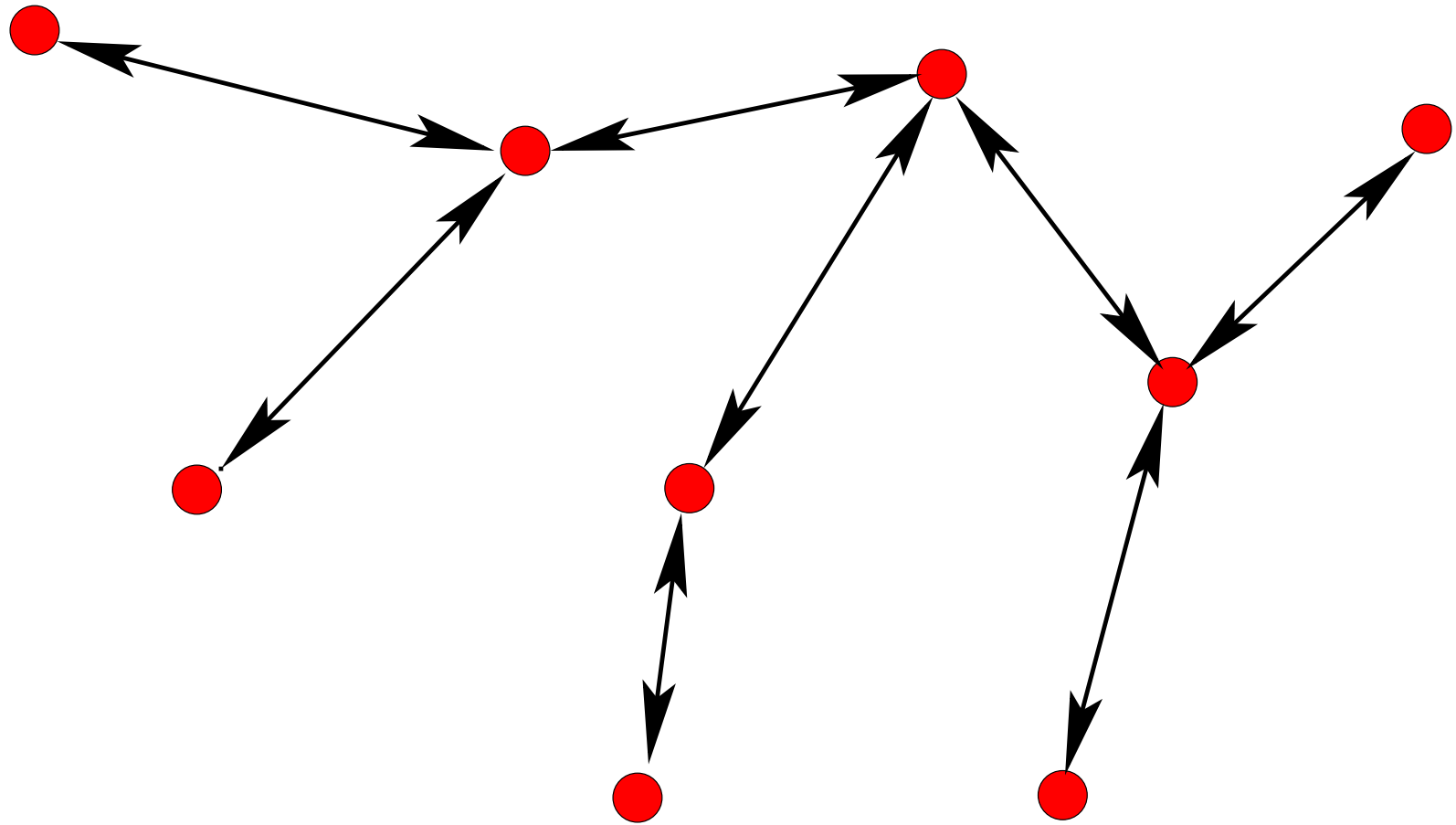


Summary of Development Process

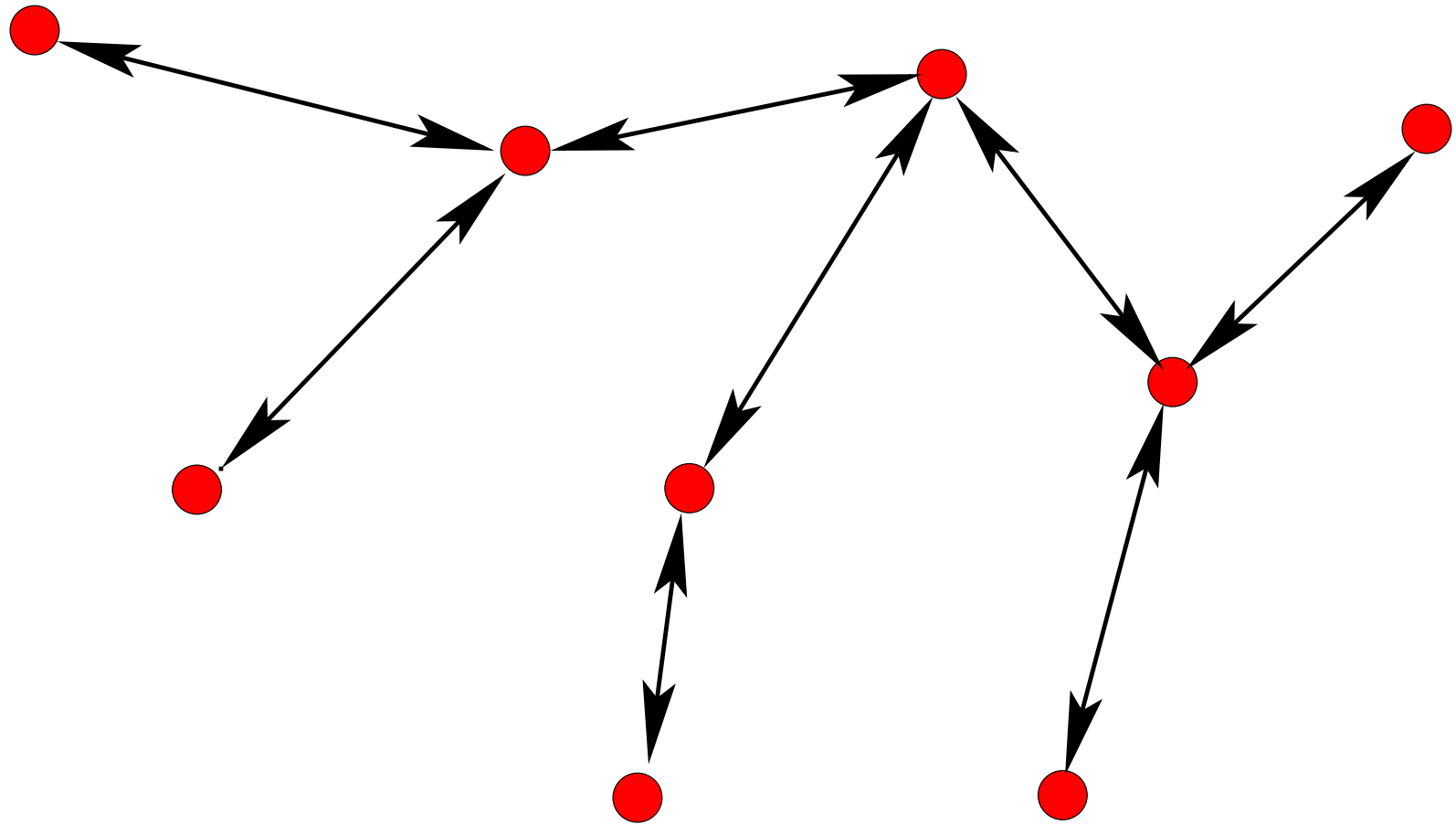
- Formal **definition** and **properties** of the network
- A **one-shot** abstract model of the protocol
- Presenting a (still abstract) loop-like **centralized solution**
- Introducing **message passing** between the nodes (**delays**)
- Modifying the data structure in order to **distribute the protocol**



Let ND be a set of nodes (with at least 2 nodes)



Let gr be a graph built and defined on ND



gr is a symmetric and irreflexive graph

g is a graph built on N

$$g \subseteq N \times N$$

g is a graph built on N

$$g \subseteq N \times N$$

g is symmetric

$$g = g^{-1}$$

g is a graph built on N

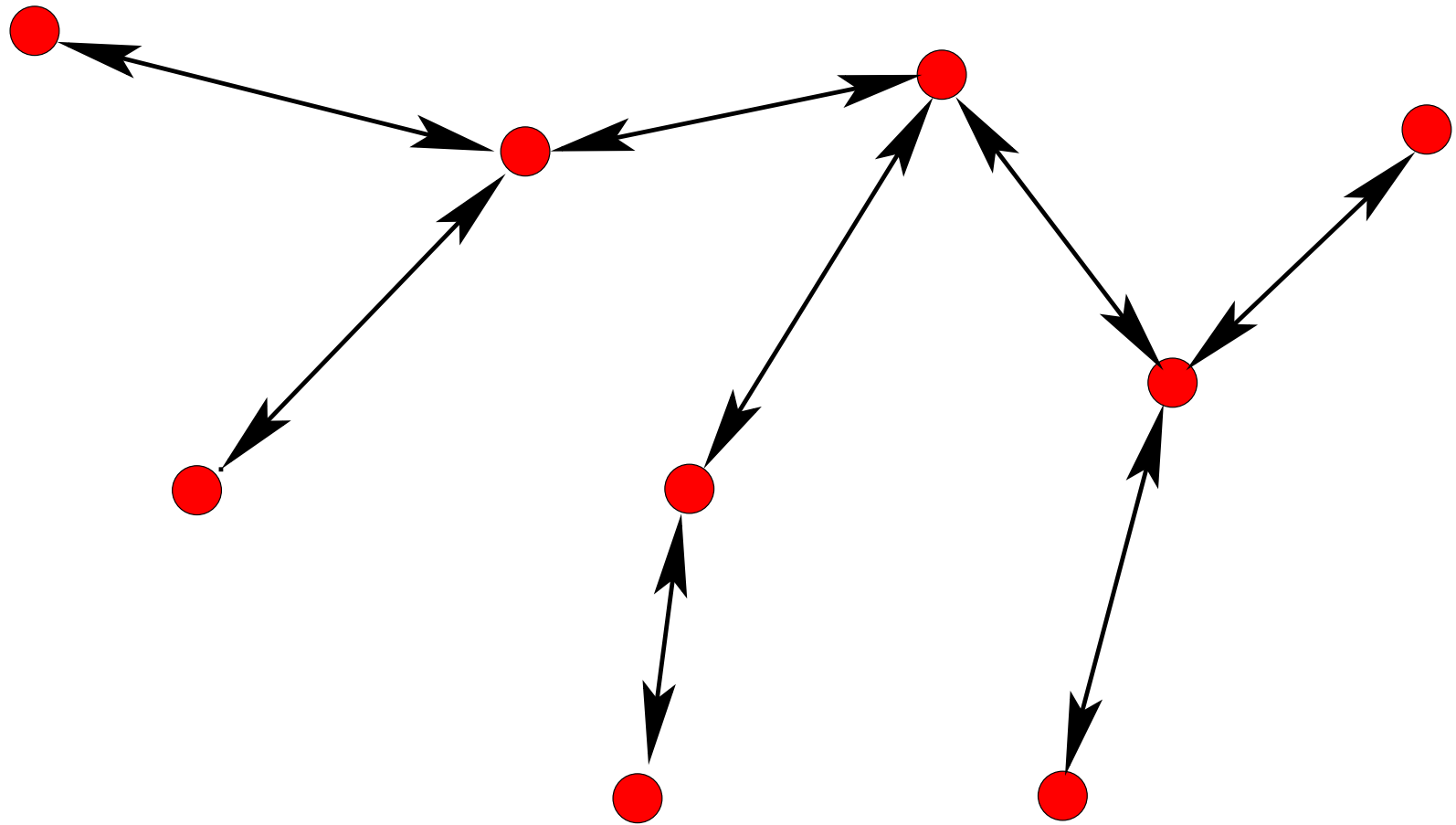
$$g \subseteq N \times N$$

g is symmetric

$$g = g^{-1}$$

g is irreflexive

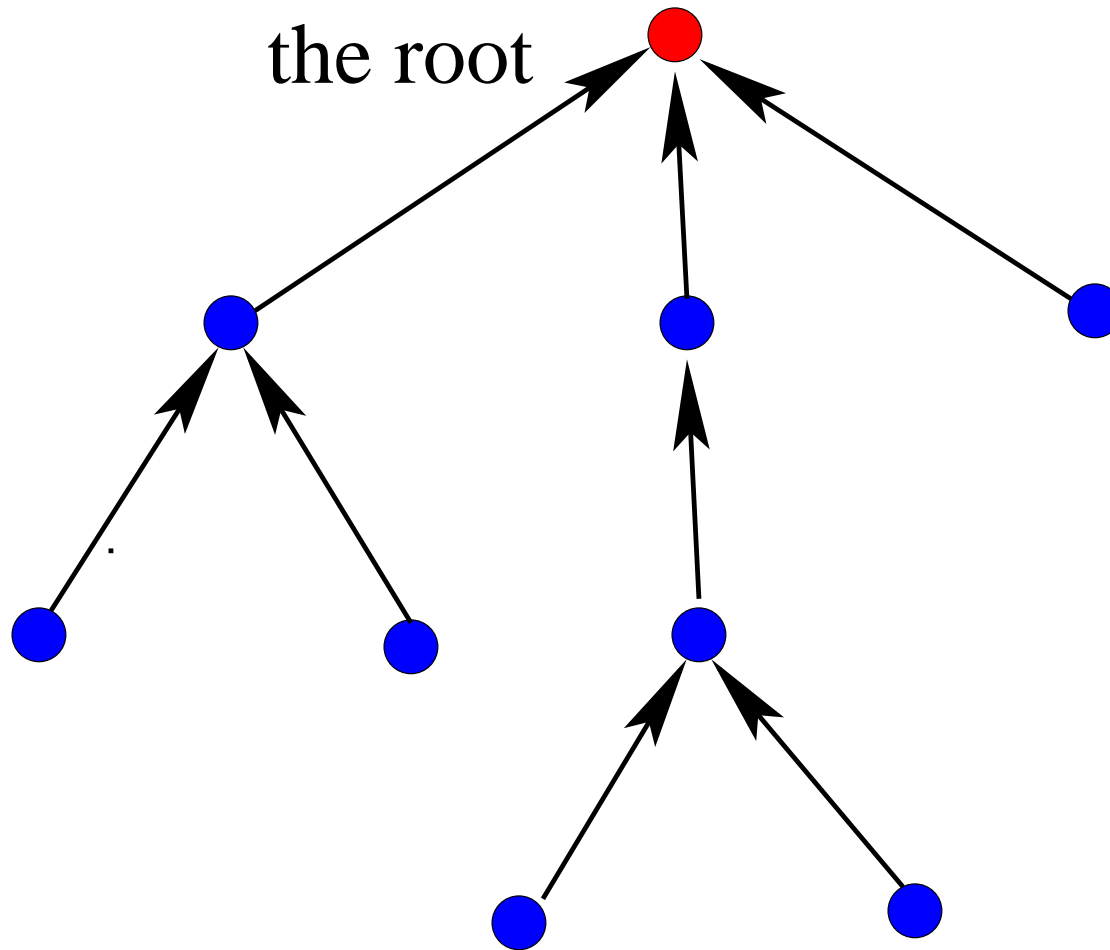
$$\text{id}(N) \cap g = \emptyset$$



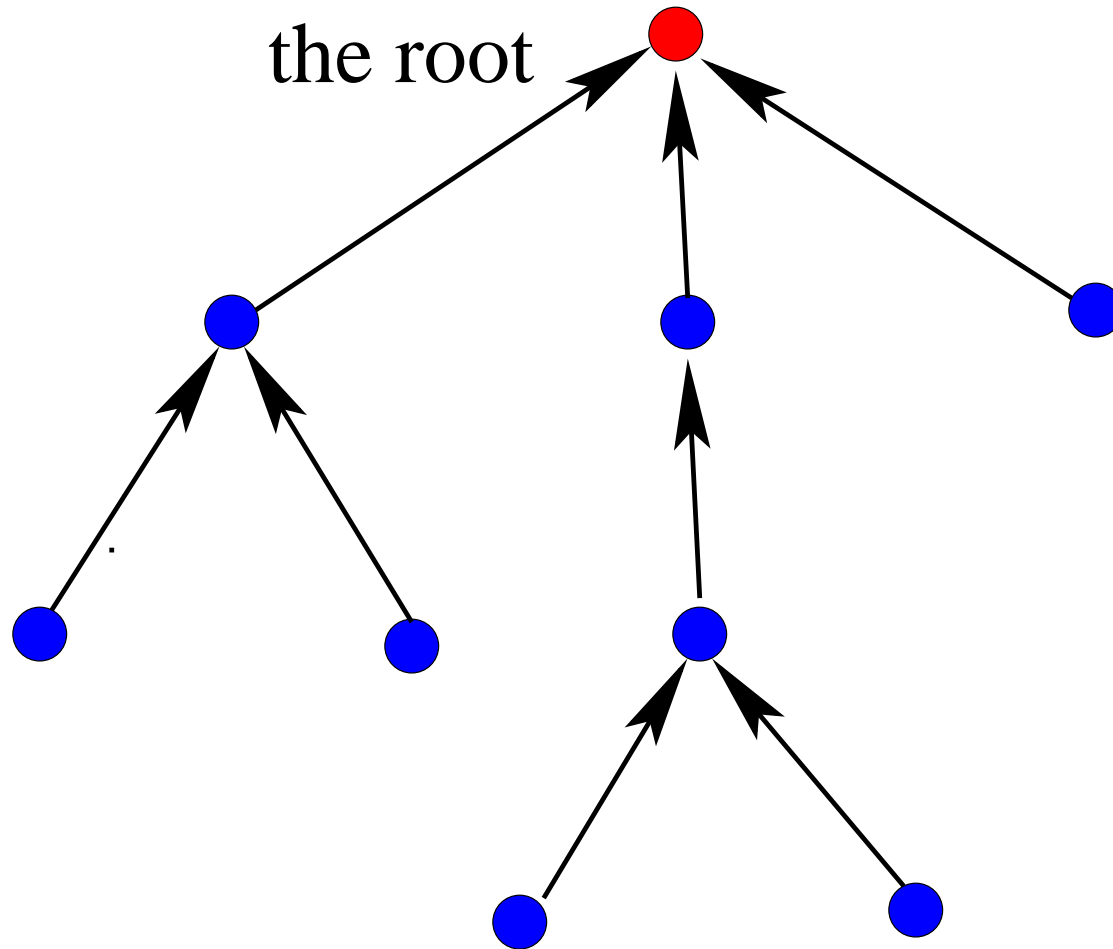
gr is connected and acyclic

A Little Detour Through Trees

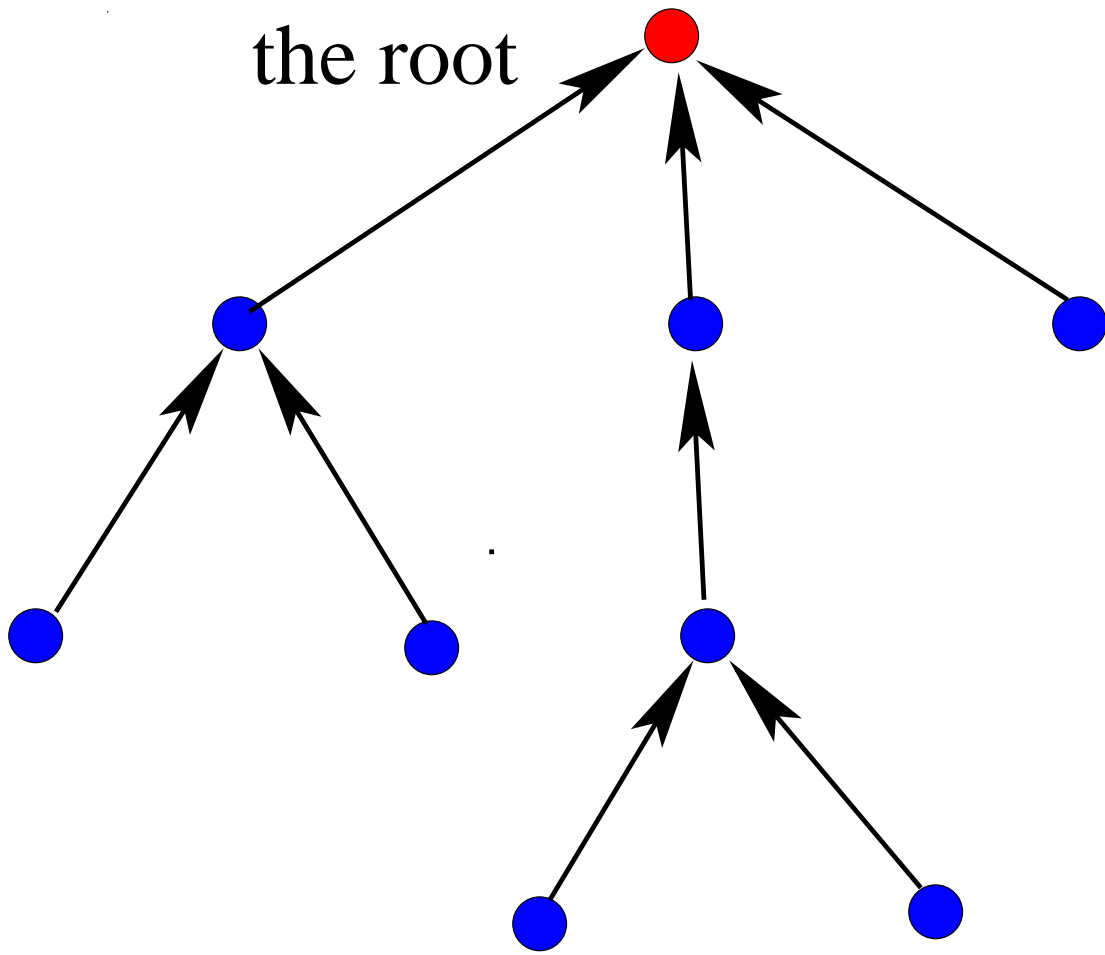
- A tree is a **special graph**
- A tree has a **root**
- A tree has a, so-called, **father function**
- A tree is **acyclic**
- A tree is **connected from the root**



A tree t built on a set of nodes

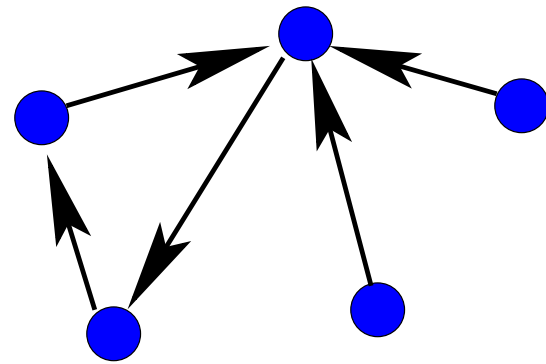


t is a function defined on ND except at the root



the root

Avoiding cycles



BAD

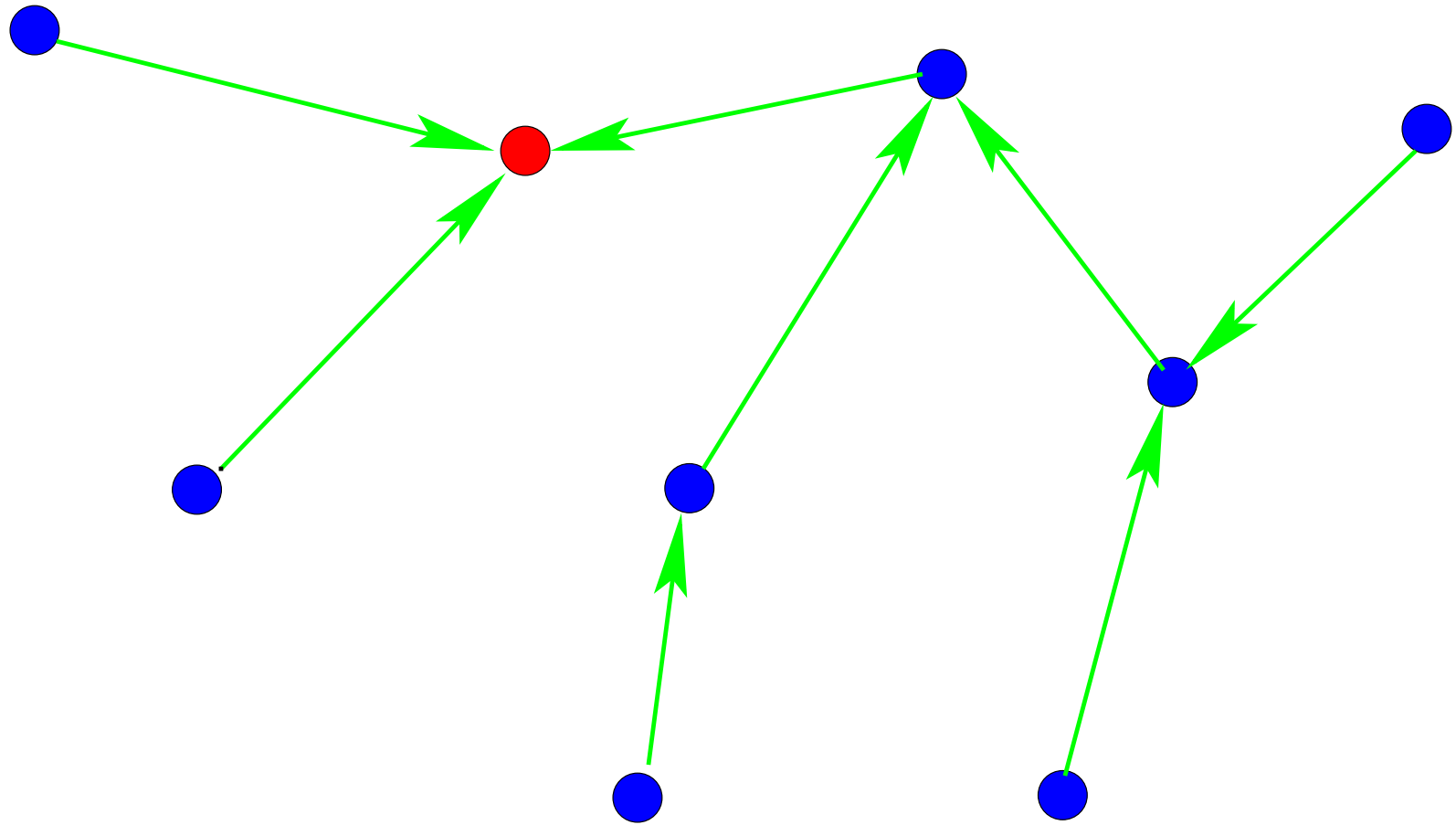
The predicate $\text{tree}(r, t)$

r is a member of N $r \in N$

t is a function $t \in N - \{r\} \rightarrow N$

t is acyclic

$$\forall q \cdot \left(\begin{array}{l} q \subseteq N \wedge \\ r \in q \wedge \\ t^{-1}[q] \subseteq q \\ \Rightarrow \\ N \subseteq q \end{array} \right)$$



A spanning tree t of the graph gr

The predicate **spanning** (r, t, g)

r, t is a tree

$\text{tree}(r, t)$

t is included in g

$t \subseteq g$

The graph g is connected and acyclic (1)

- Defining a **relation** f linking a **node** to the possible **spanning trees** of g having **that node as a root**:

$$f \in N \rightarrow (N \leftrightarrow N)$$

$$\forall(x, t) \cdot \left(\begin{array}{l} x \in N \wedge \\ \text{spanning}(x, t, g) \\ \Rightarrow \\ t = f(x) \end{array} \right) \quad \forall x \cdot \left(\begin{array}{l} x \in N \wedge \\ \Rightarrow \\ \text{spanning}(x, f(x), g) \end{array} \right)$$

Election in One Shot: Building a Spanning Tree

- Variables l and s

$$l \in N$$

$$s \in N \leftrightarrow N$$

elect $\hat{=}$

any x **where**

$$x \in N$$

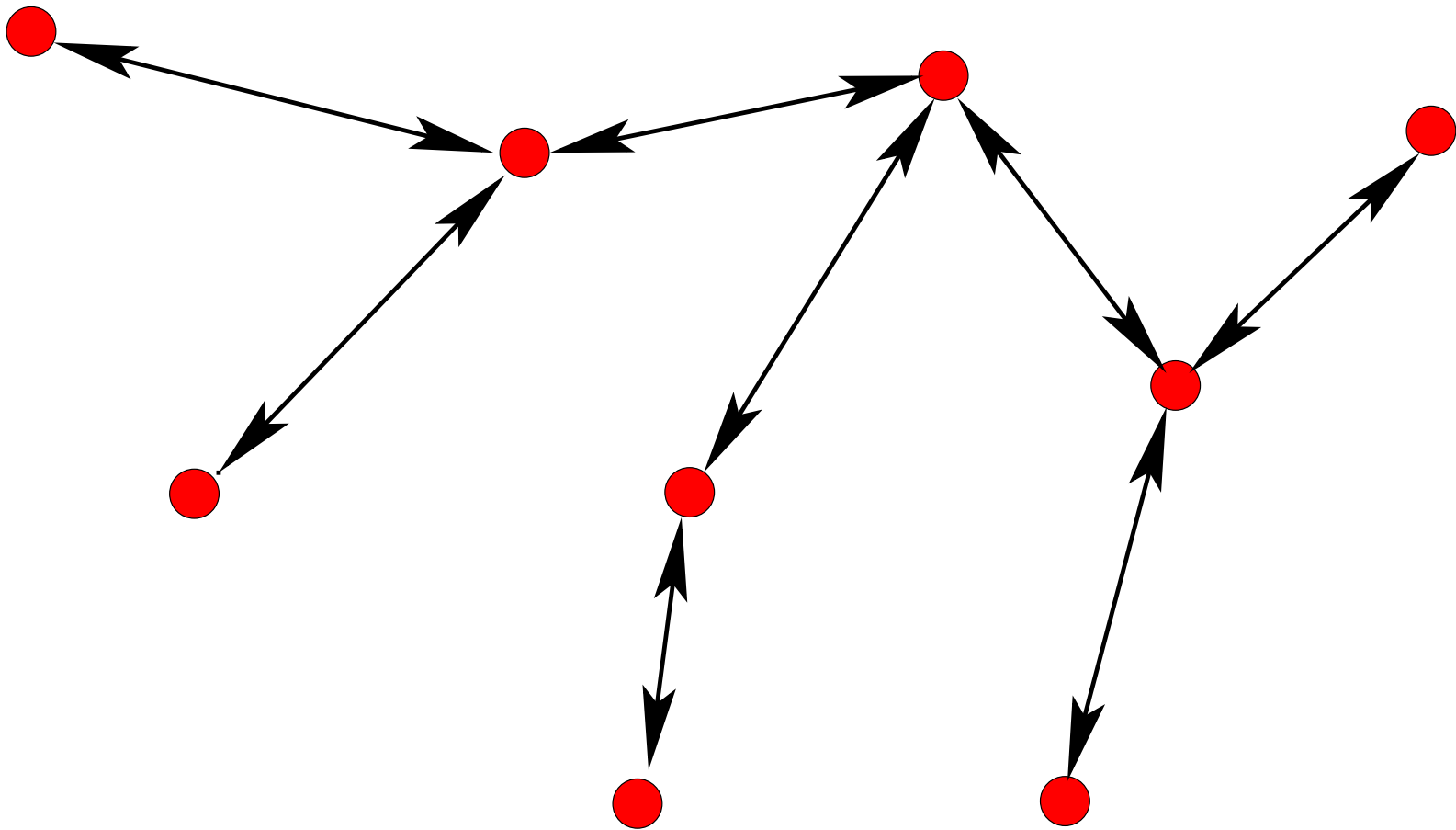
then

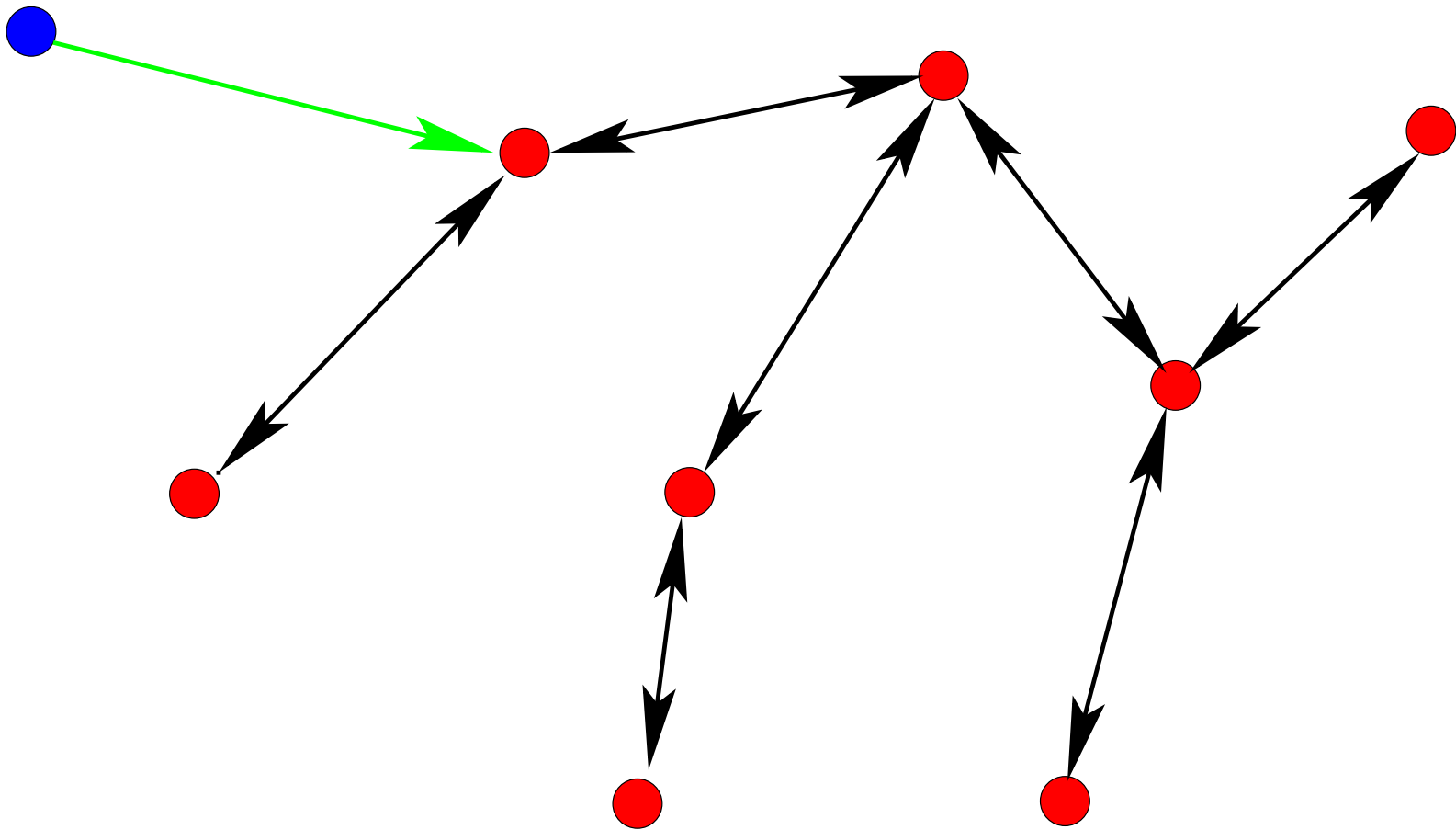
$$l, s := x, f(x)$$

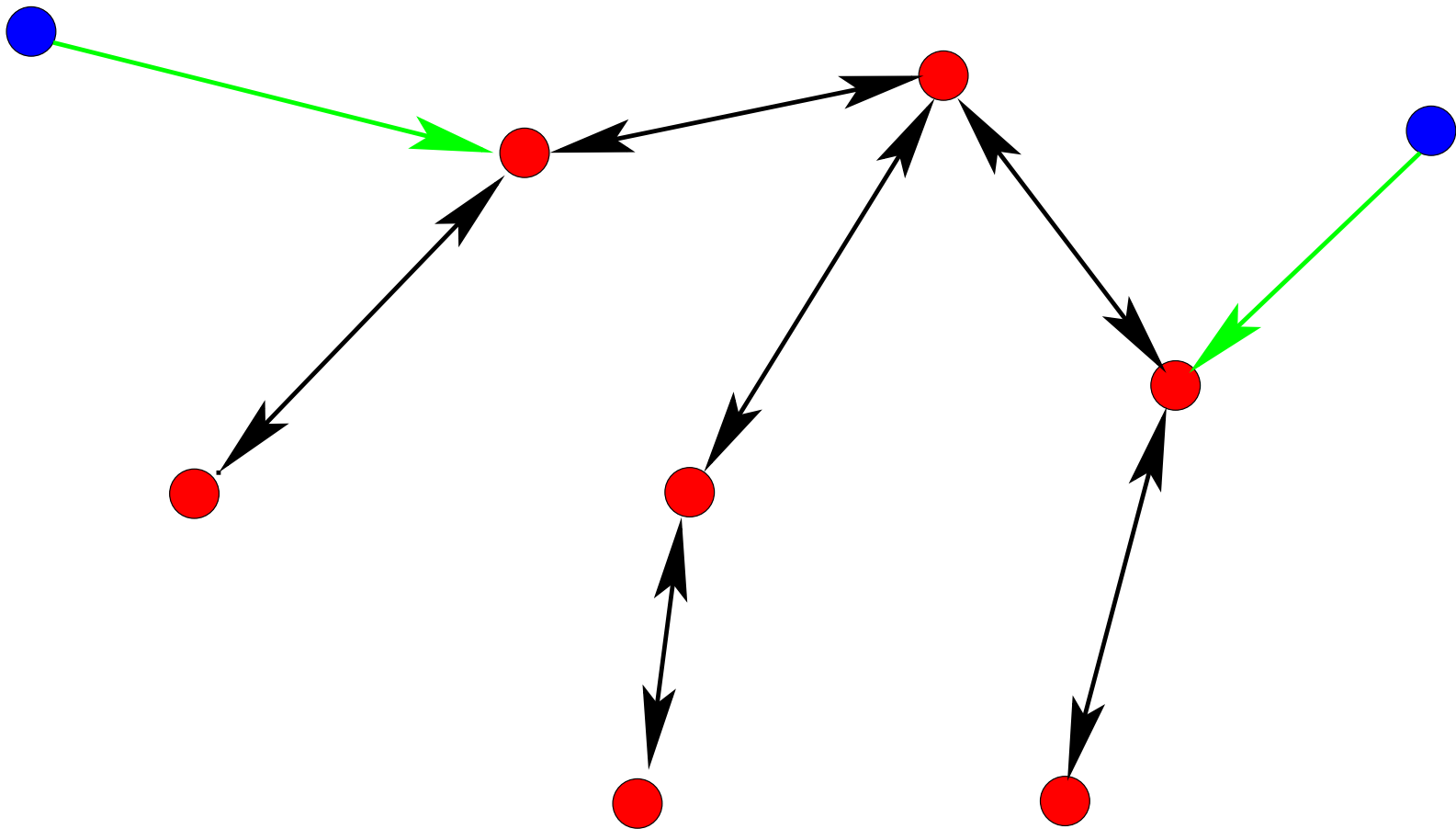
end

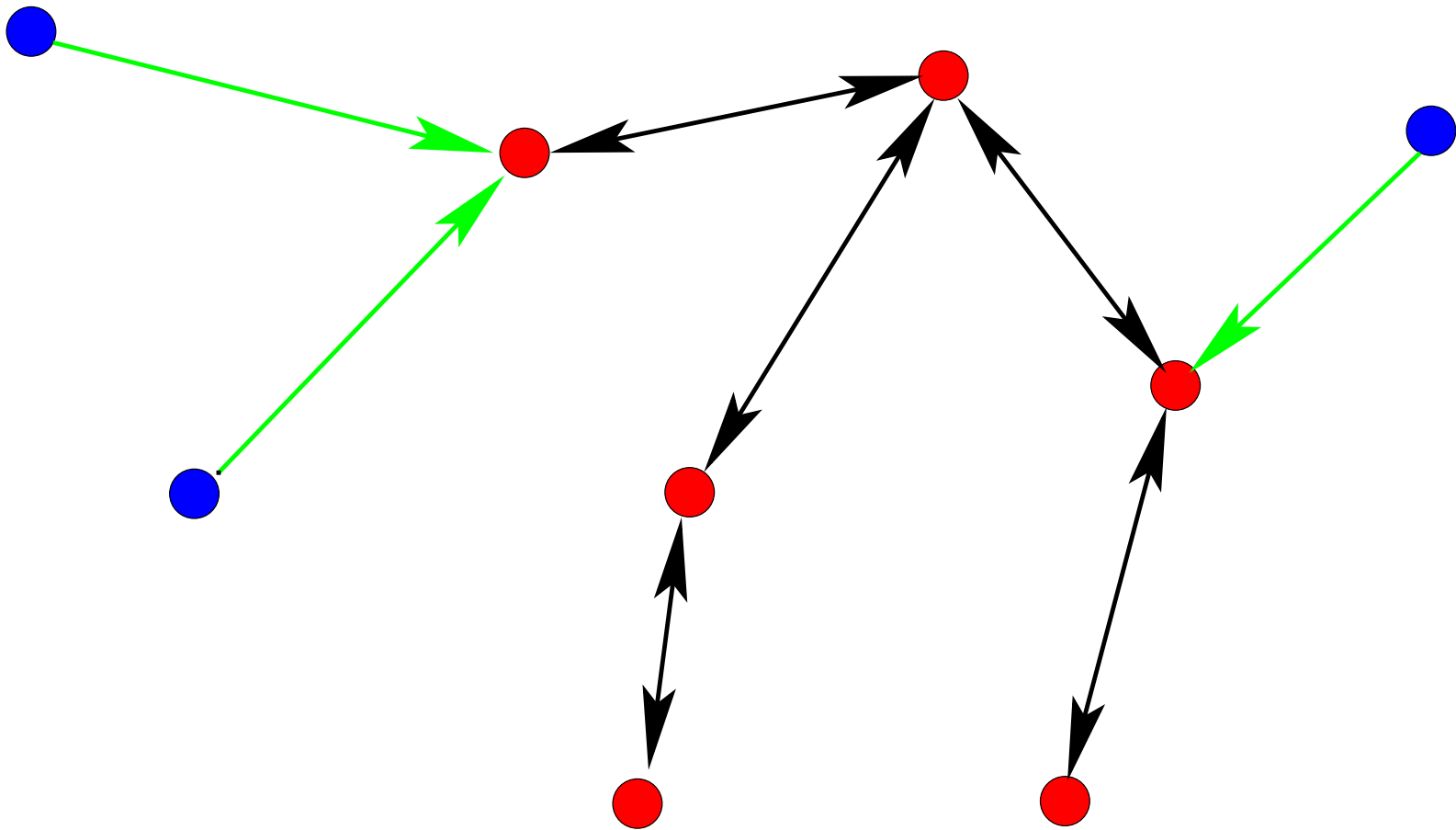
First Refinement (1)

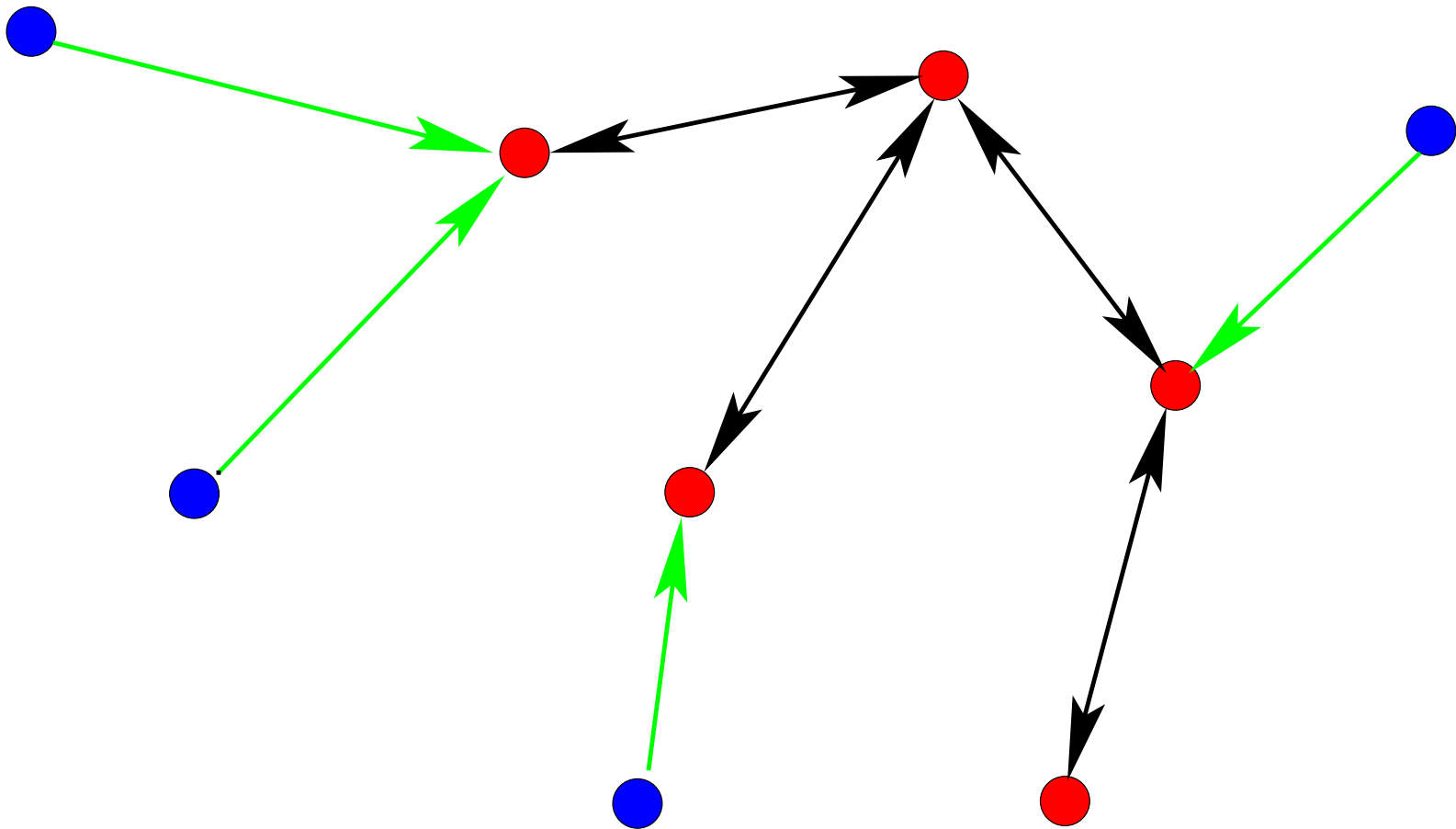
- Introducing a **new variable**, t , corresponding to the **"tree" in construction**
- Introducing a **new event**: the **progression event**
- Defining the **invariant**
- Back to the **animation** : Observe the construction of the tree

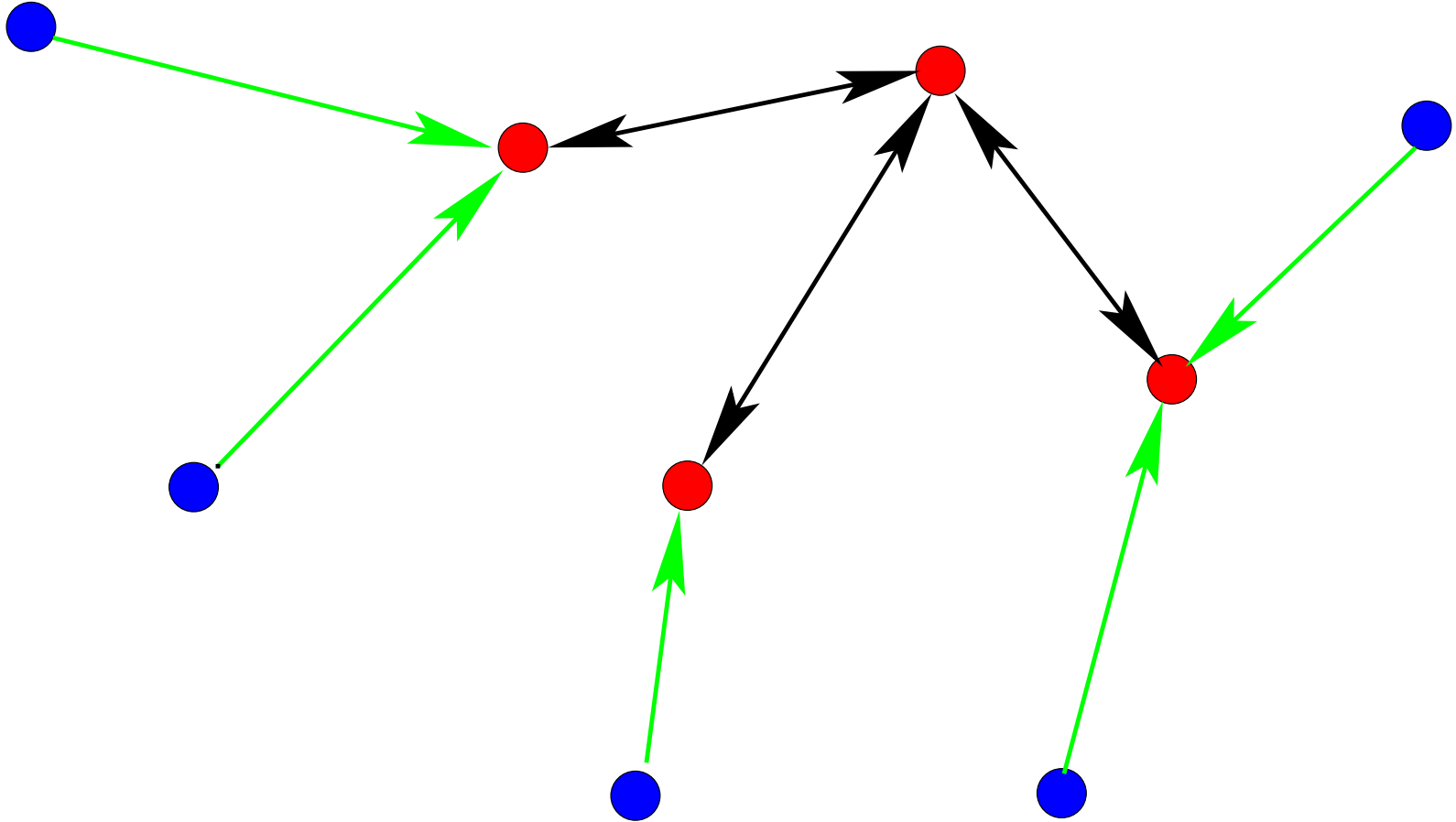


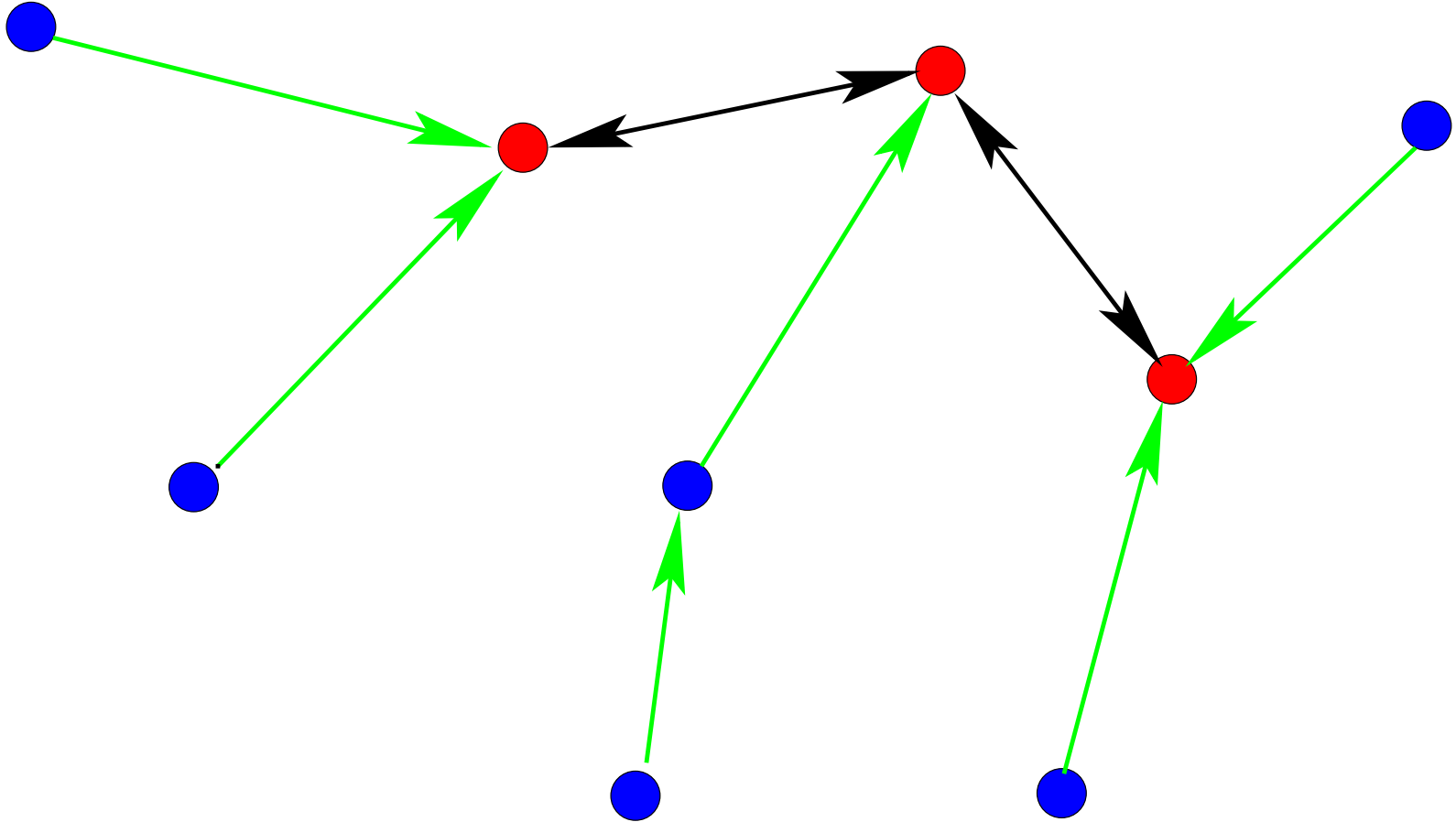


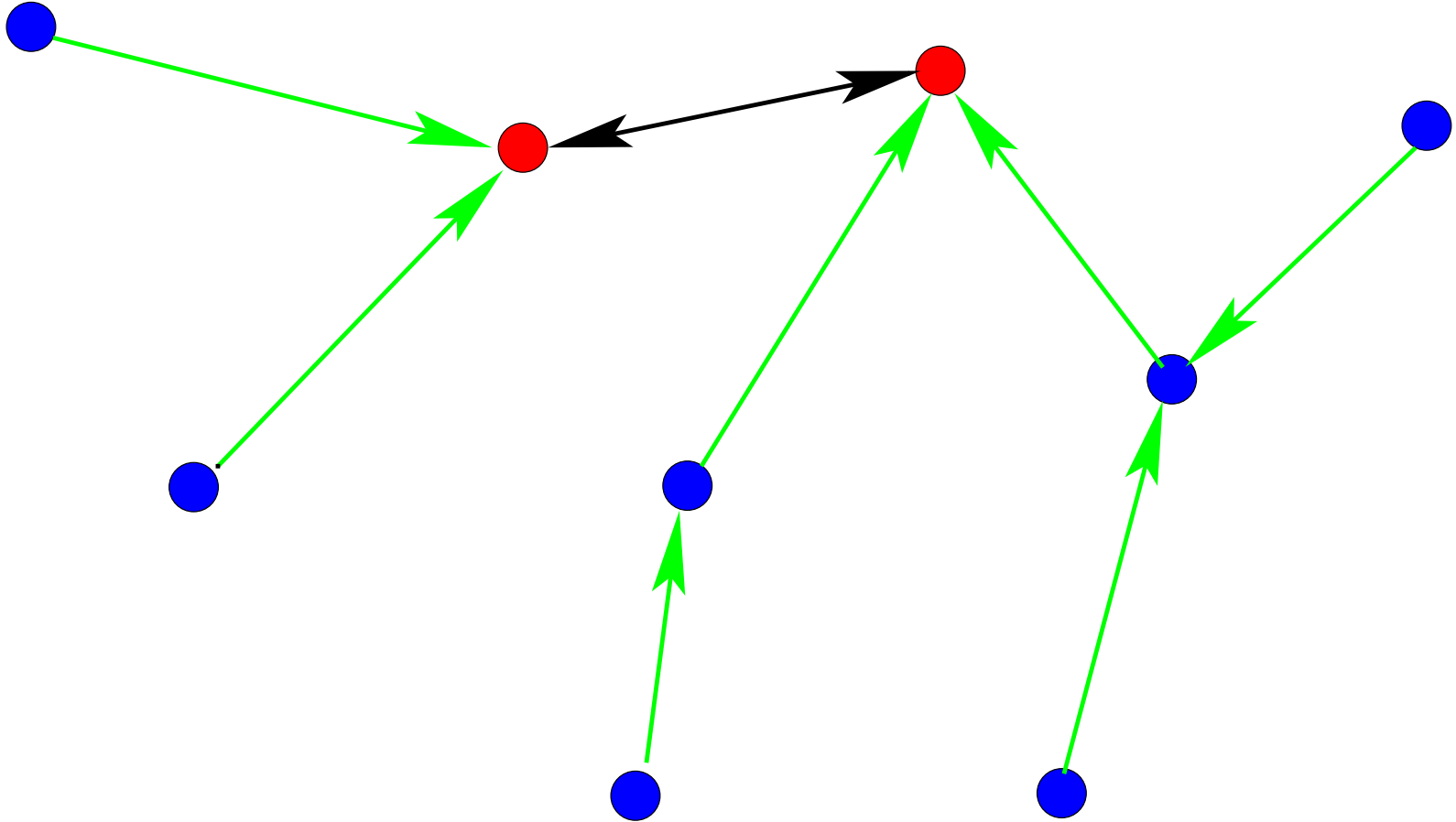


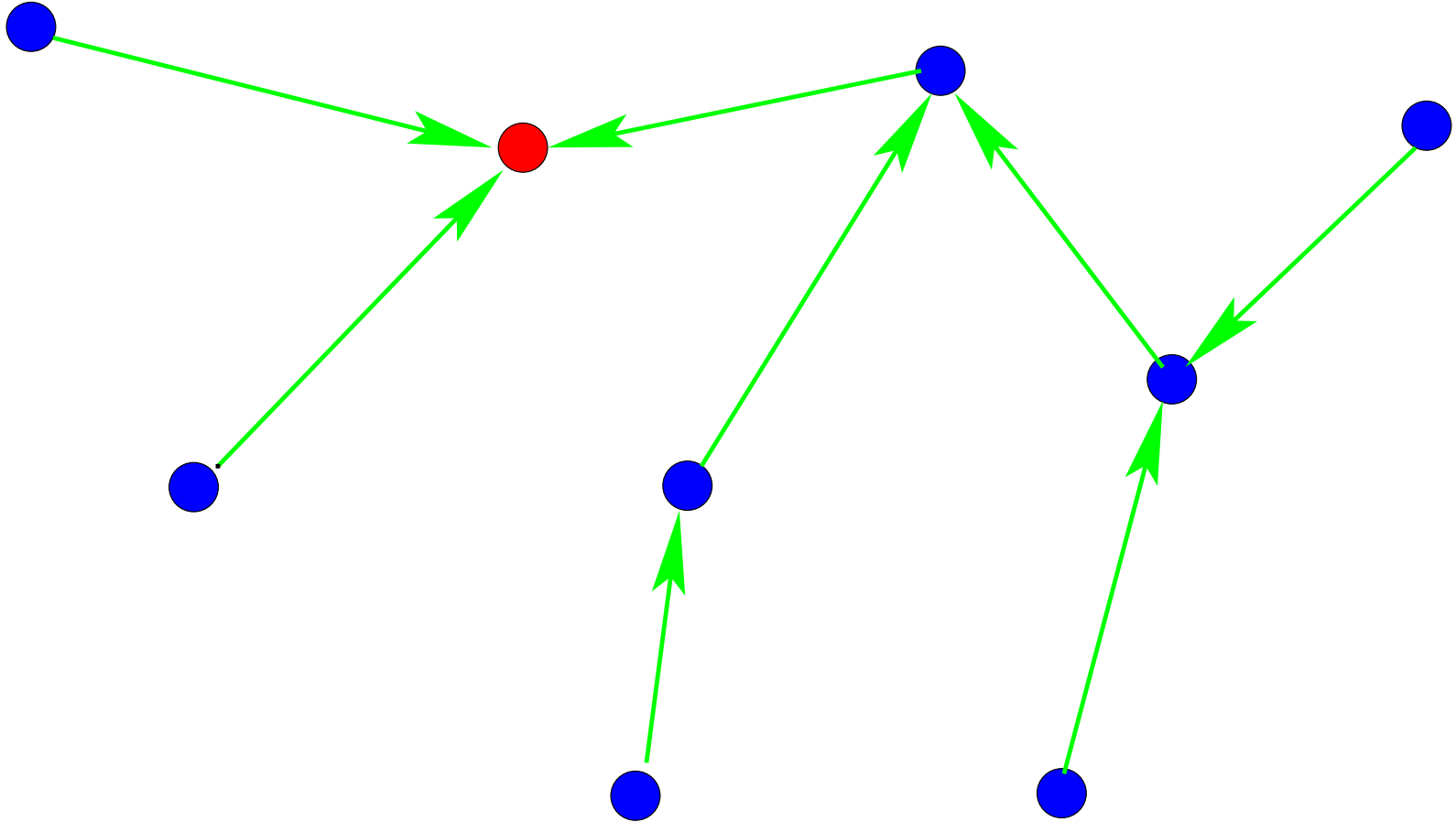












- The **green arrows** correspond to the t function
- The **blue nodes** are the **domain** of t
- The function t is a **forest (multi-tree)** on nodes
- The **red nodes** are the **roots** of these trees

The predicate **invariant** (t)

$$t \in N \leftrightarrow N$$

The predicate **invariant** (t)

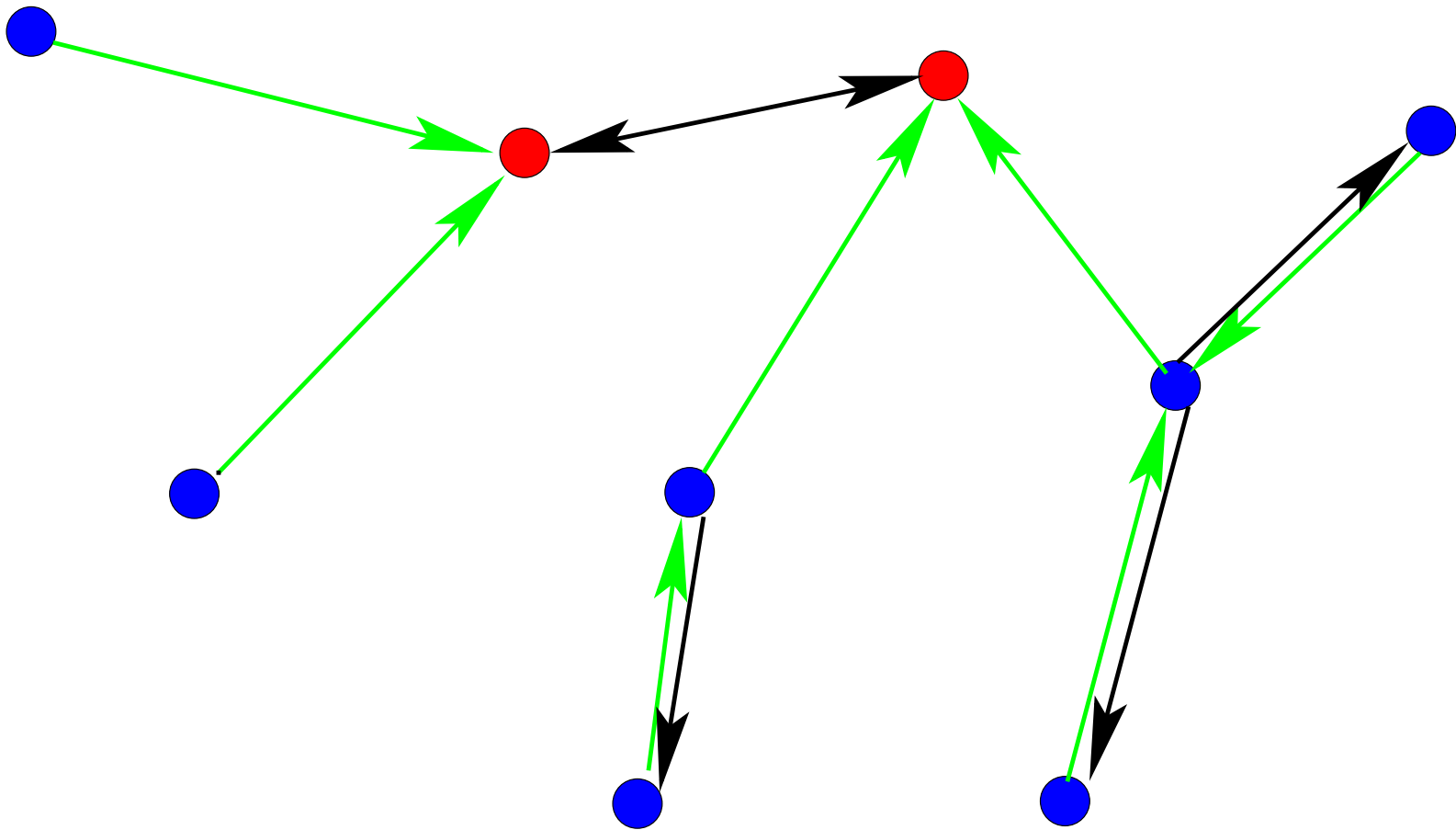
$$t \in N \leftrightarrow N$$

$$\forall p \cdot \left(\begin{array}{l} p \subseteq N \quad \wedge \\ N - \text{dom}(t) \subseteq p \quad \wedge \\ t^{-1}[p] \subseteq p \\ \Rightarrow \\ N \subseteq p \end{array} \right)$$

The predicate **invariant** (t)

$$t \in N \leftrightarrow N$$

$$\text{dom}(t) \triangleleft (t \cup t^{-1}) = \text{dom}(t) \triangleleft g$$



The predicate **invariant** (t)

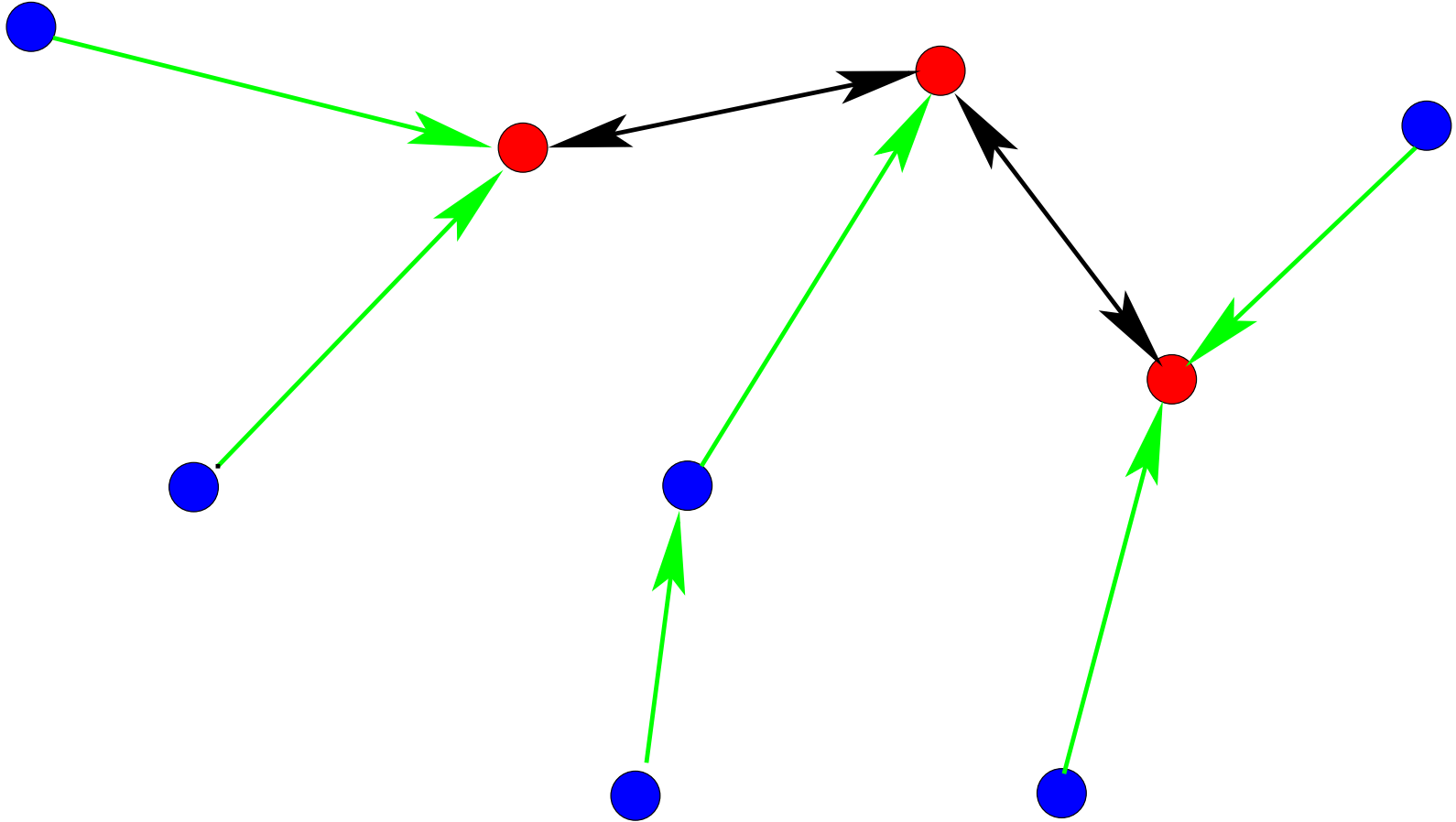
$$t \in N \leftrightarrow N$$

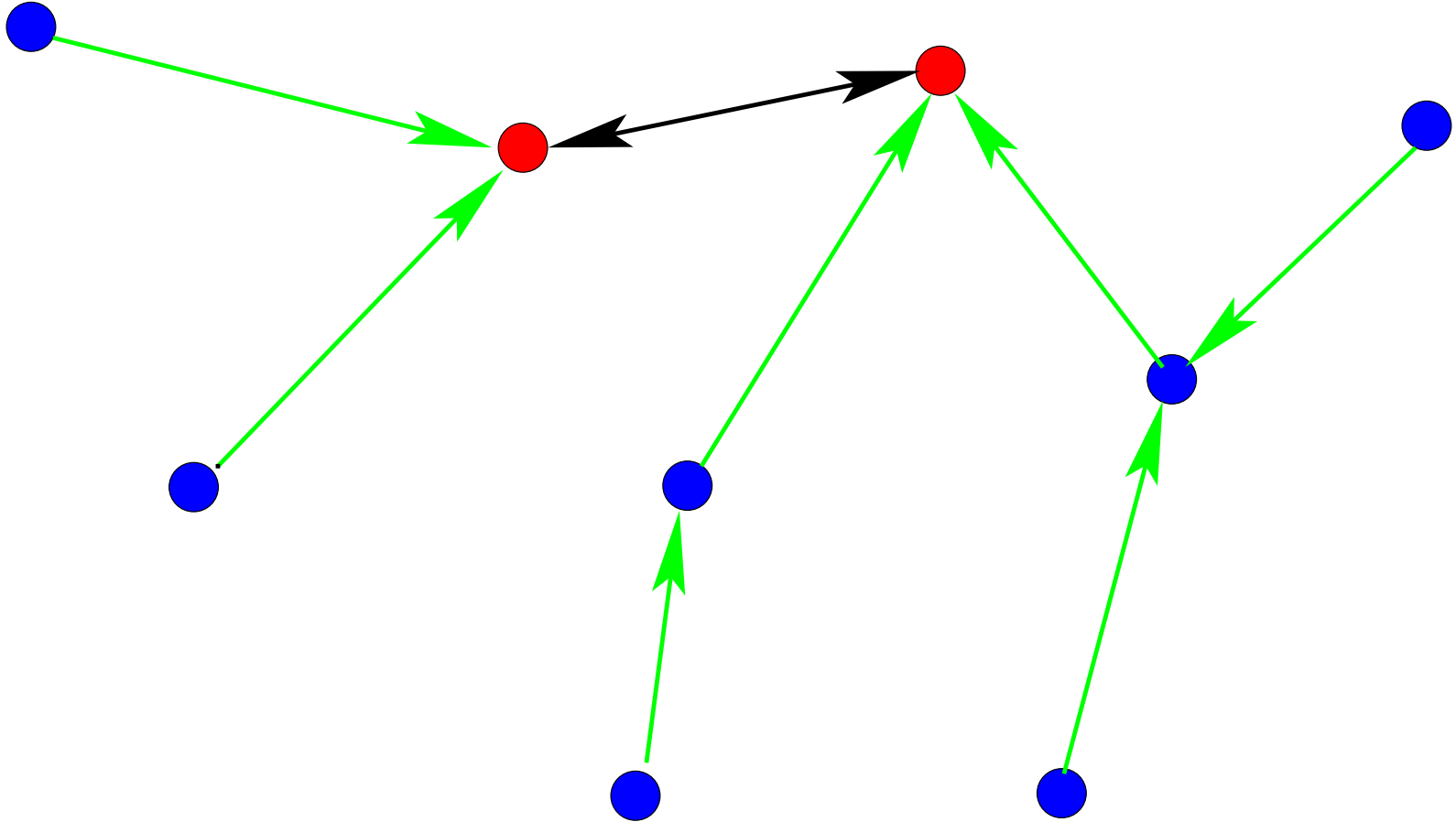
$$\text{dom}(t) \triangleleft (t \cup t^{-1}) = \text{dom}(t) \triangleleft g$$

$$t \cap t^{-1} = \emptyset$$

First Refinement (2)

- Introducing the **new event** "progress"
- Refining the abstract event "elect"
- Back to the **animation** : Observe the "guard" of progress





When a red node x is connected to **AT MOST** one other red node y then event "progress" can take place

progress $\hat{=}$

any x, y **where**

$$x \mapsto y \in g \wedge$$

$$x \notin \text{dom}(t) \wedge$$

$$y \notin \text{dom}(t) \wedge$$

$$g[\{x\}] = t^{-1}[\{x\}] \cup \{y\}$$

then

$$t := t \cup \{x \mapsto y\}$$

end

To be proved

$$\text{invariant}(t) \quad \wedge$$

$$x \mapsto y \in g \quad \wedge$$

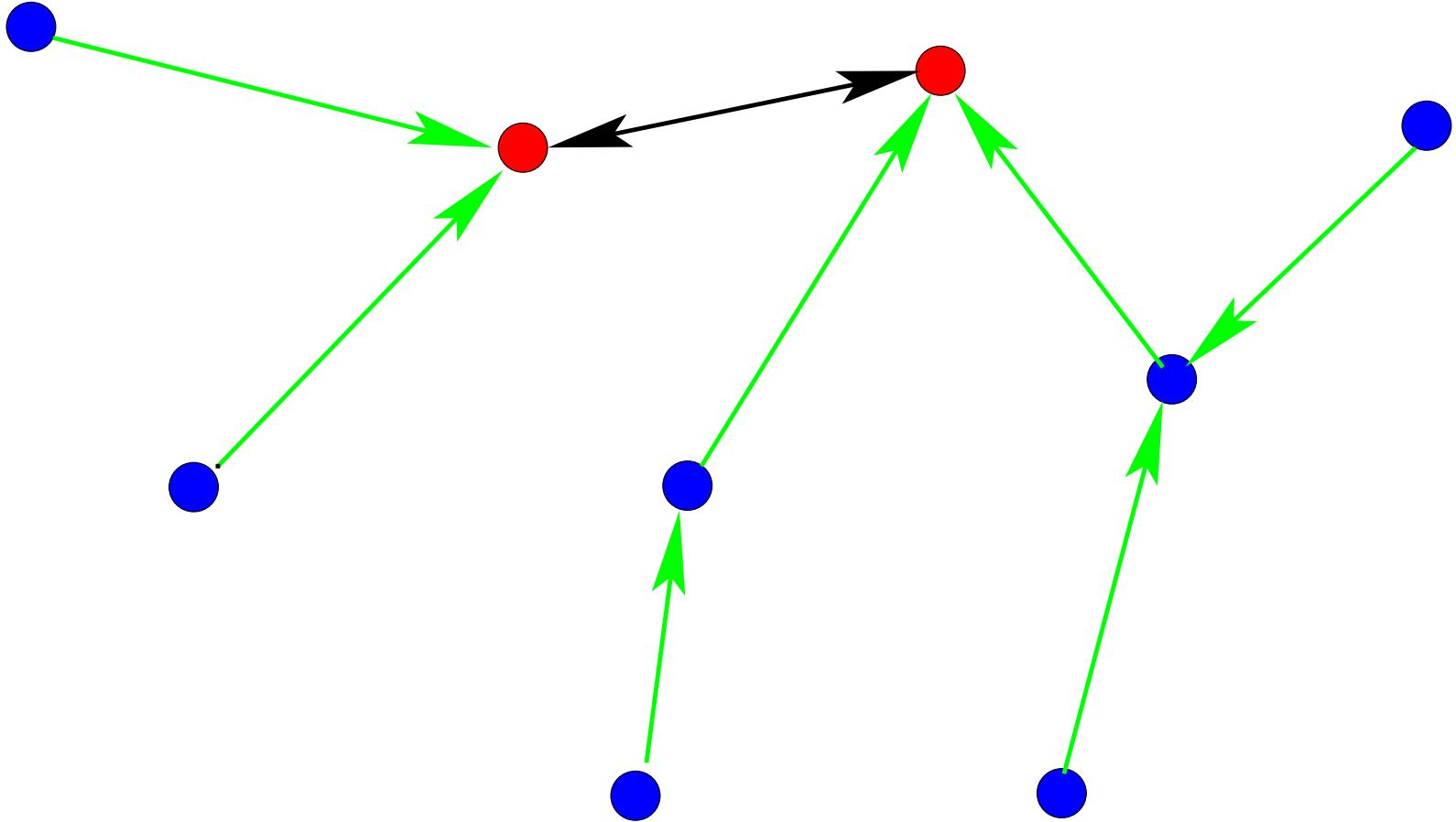
$$x \notin t \quad \wedge$$

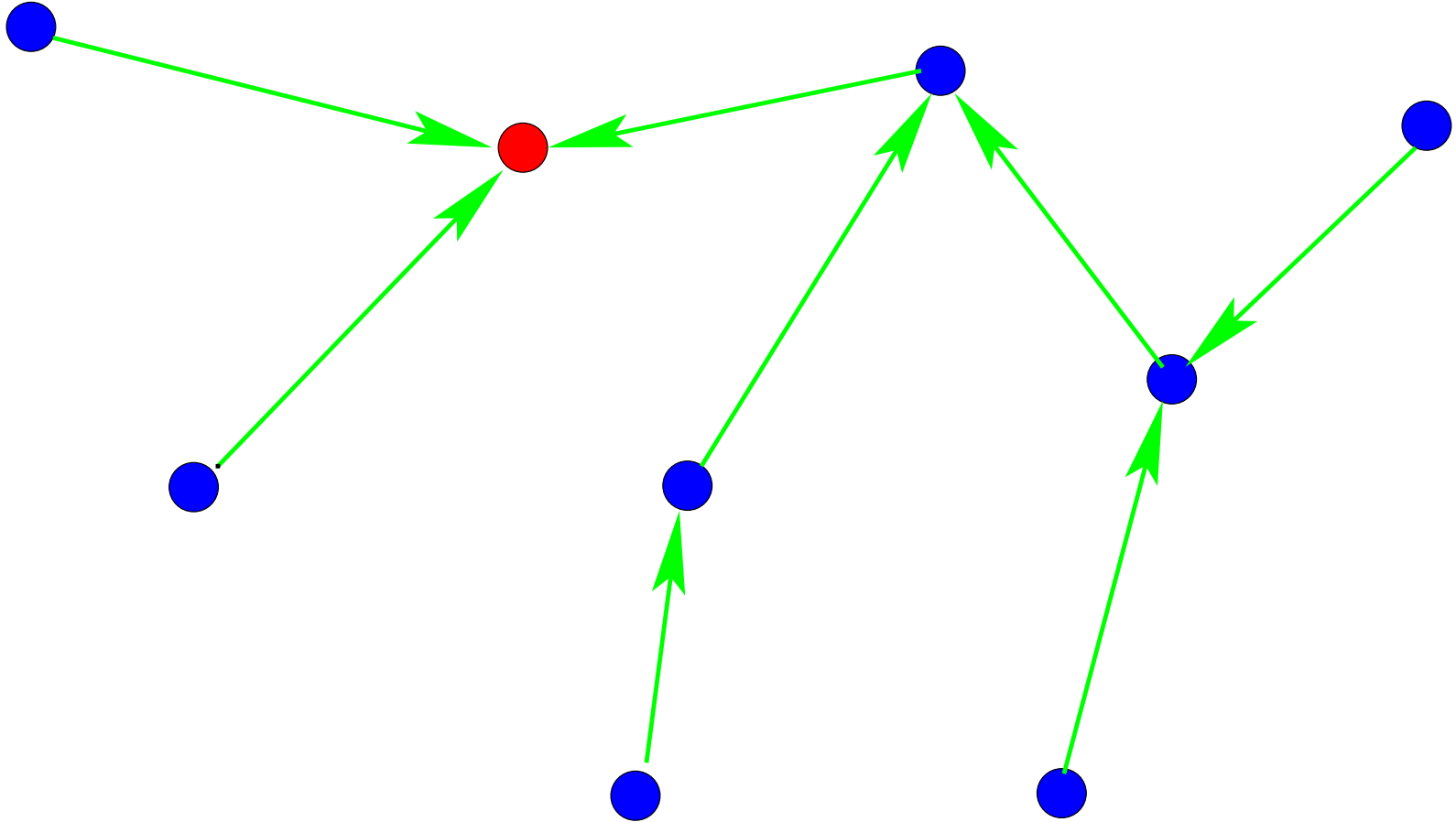
$$y \notin t \quad \wedge$$

$$g[\{x\}] = t^{-1}[\{x\}] \cup \{y\}$$

\Rightarrow

$$\text{invariant}(t \cup \{x \mapsto y\})$$





When a **red node** x is **ONLY** connected to **blue nodes** then event "elect" can take place

elect $\hat{=}$

any x **where**

$$x \in N \wedge$$

$$g[\{x\}] = t^{-1}[\{x\}]$$

then

$$l, s := x, t$$

end

elect $\hat{=}$

any x **where**

$x \in N$

then

$l, s := x, f(x)$

end

elect $\hat{=}$

any x **where**

$x \in N \wedge$

$g[\{x\}] = t^{-1}[\{x\}]$

then

$l, s := x, t$

end

To be proved

$$\begin{aligned} & \text{invariant}(t) \quad \wedge \\ & x \in N \quad \wedge \\ & g[\{x\}] = t^{-1}[\{x\}] \\ \Rightarrow \\ & t = f(x) \end{aligned}$$

An idea

$$x \notin \text{dom}(t)$$

$$N \subseteq \{n \mid n \in N - \{x\} \wedge n, f(x)(n) \in t\} \cup \{x\}$$

Summary of First Refinement

- 7 proofs

- Among which 1 were interactive (one is a bit difficult !)

12 proofs and 2 interactives (in a previous work)

Why more proofs?

elect $\hat{=}$

begin

$l, s : \text{spanning}(l, s, g)$

end

elect $\hat{=}$

any x **where**

$x \in N$

then

$l, s := x, f(x)$

end

First Refinement: Reshaping the invariant

- Another idea?
- The previous proof is quiet difficult for an non expert
- How can we prove that at end (for the contention) there are only two nodes
-

The predicate **invariant** (t)

$$t \in N \leftrightarrow N$$

$$t \cap t^{-1} = \emptyset$$

$$\forall x \cdot \left(\begin{array}{l} x \in N - \text{dom}(t) \\ \Rightarrow \\ t \subseteq f(x) \end{array} \right)$$

$$\text{dom}(t) \triangleleft (t \cup t^{-1}) = \text{dom}(t) \triangleleft g$$

An idea

$$x \notin \text{dom}(t)$$

$$N \subseteq \text{dom}(t) \cup \{x\}$$

Summary of First Refinement

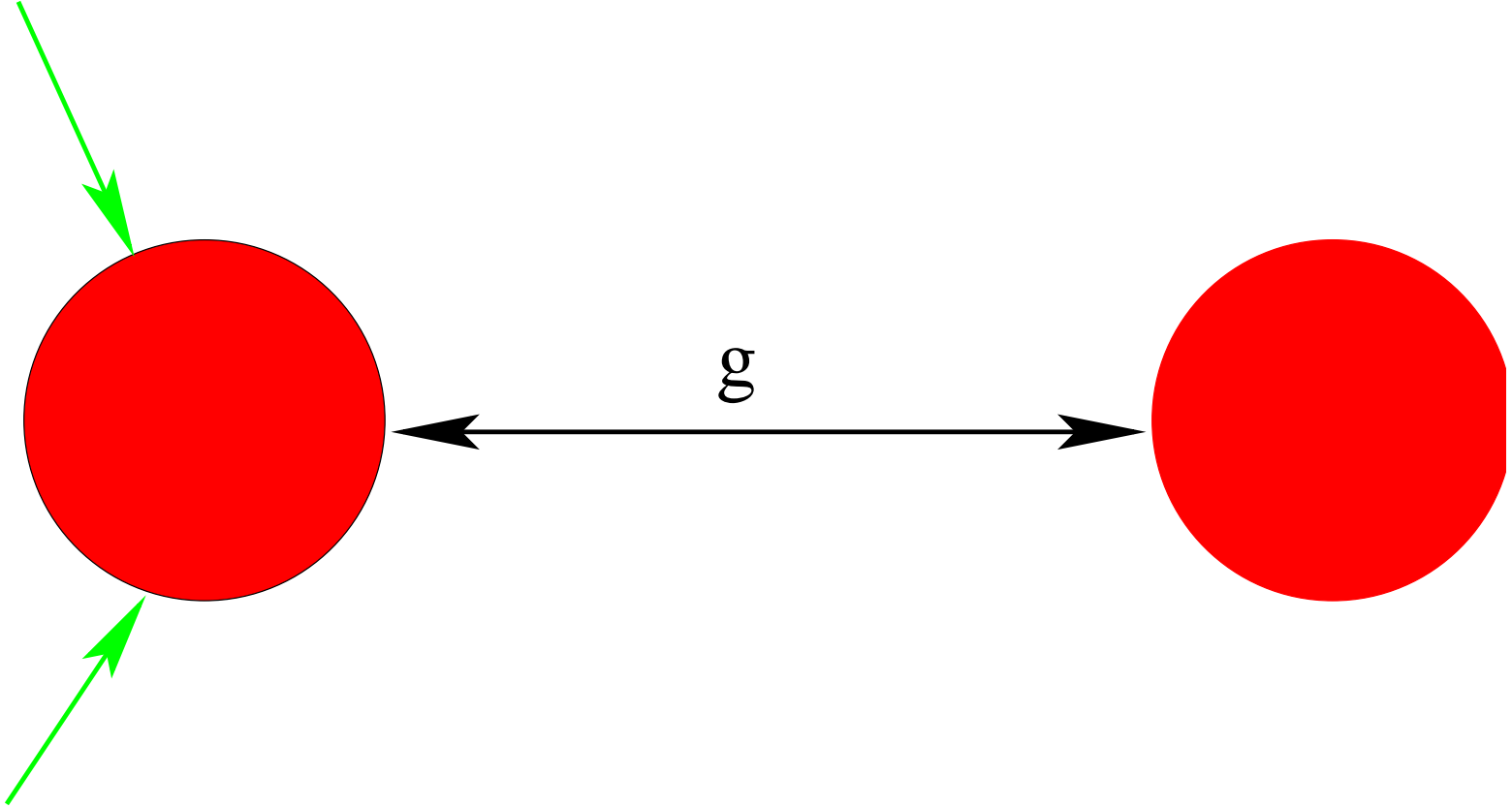
- 10 proofs
- Among which 2 were interactive

Second Refinement

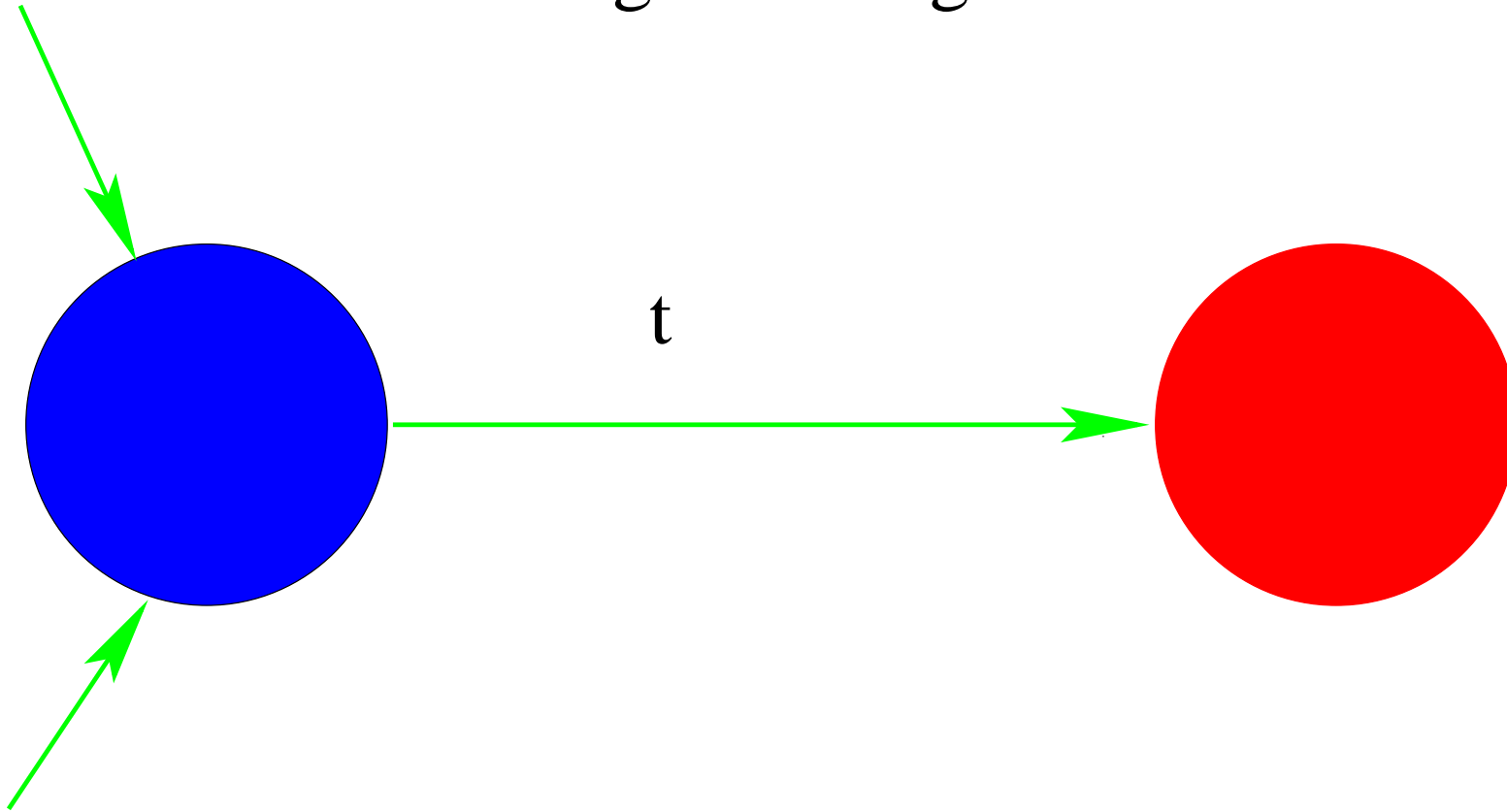
- Nodes are **communicating** with their neighbors
- This is done by means of **messages**
- Messages are **acknowledged**
- Acknowledgements are **confirmed**
- Next is a **local** animation

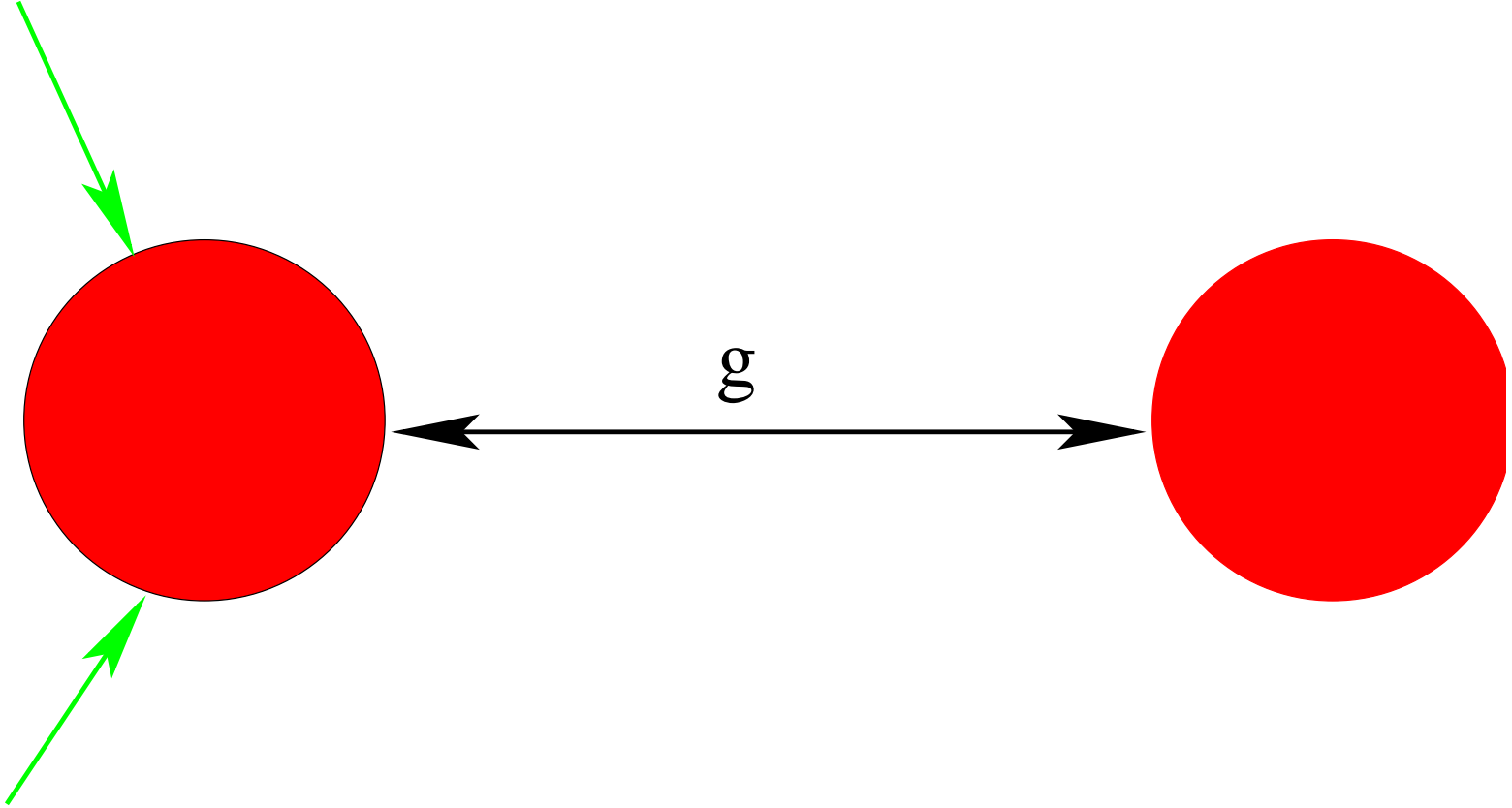
Second Refinement

- Nodes are **communicating** with their neighbors
- This is done by means of **messages**
- **No acknowledgment**
- **No confirmation**
- Next is a **local** animation

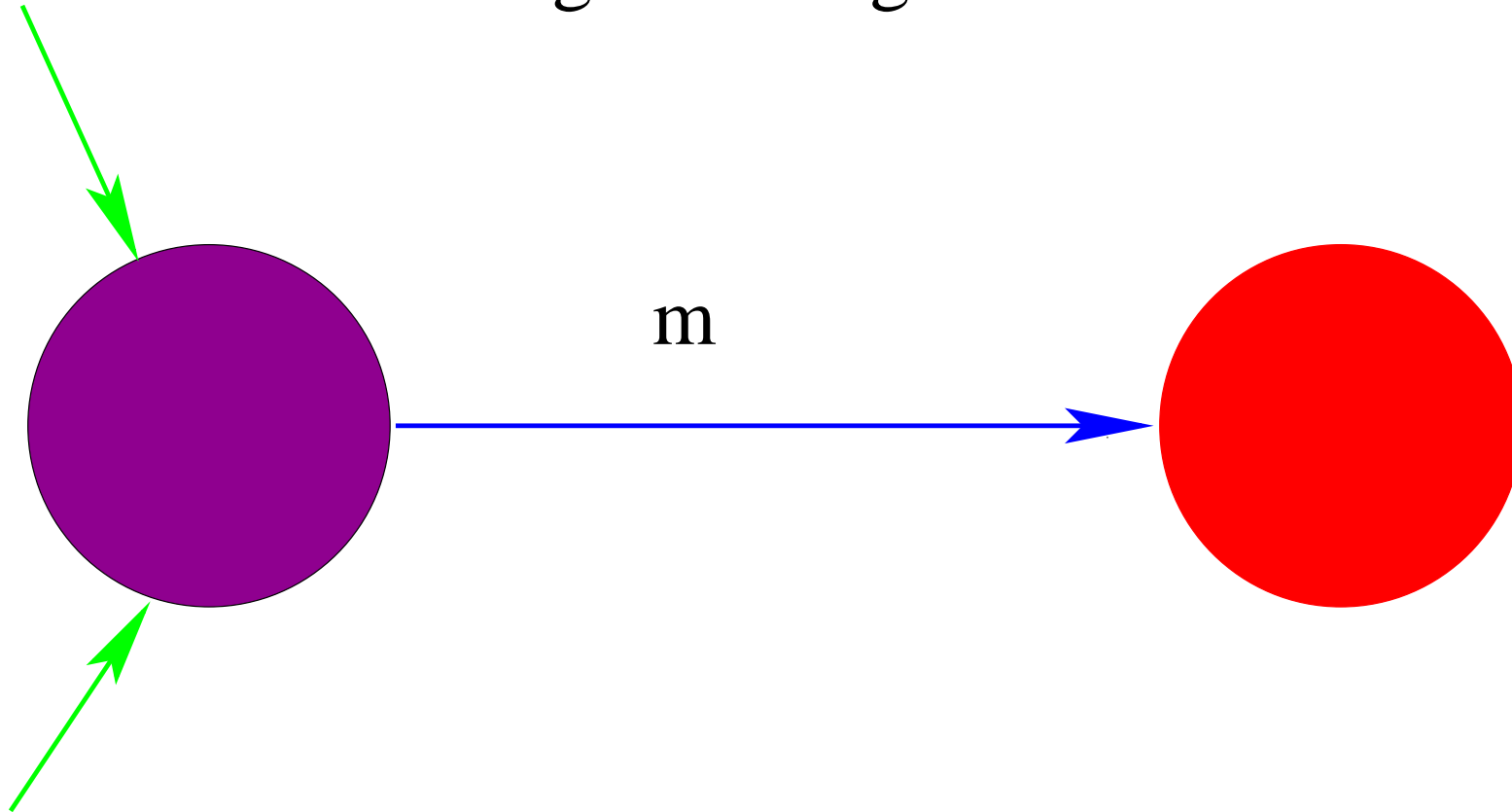


Receiving a message

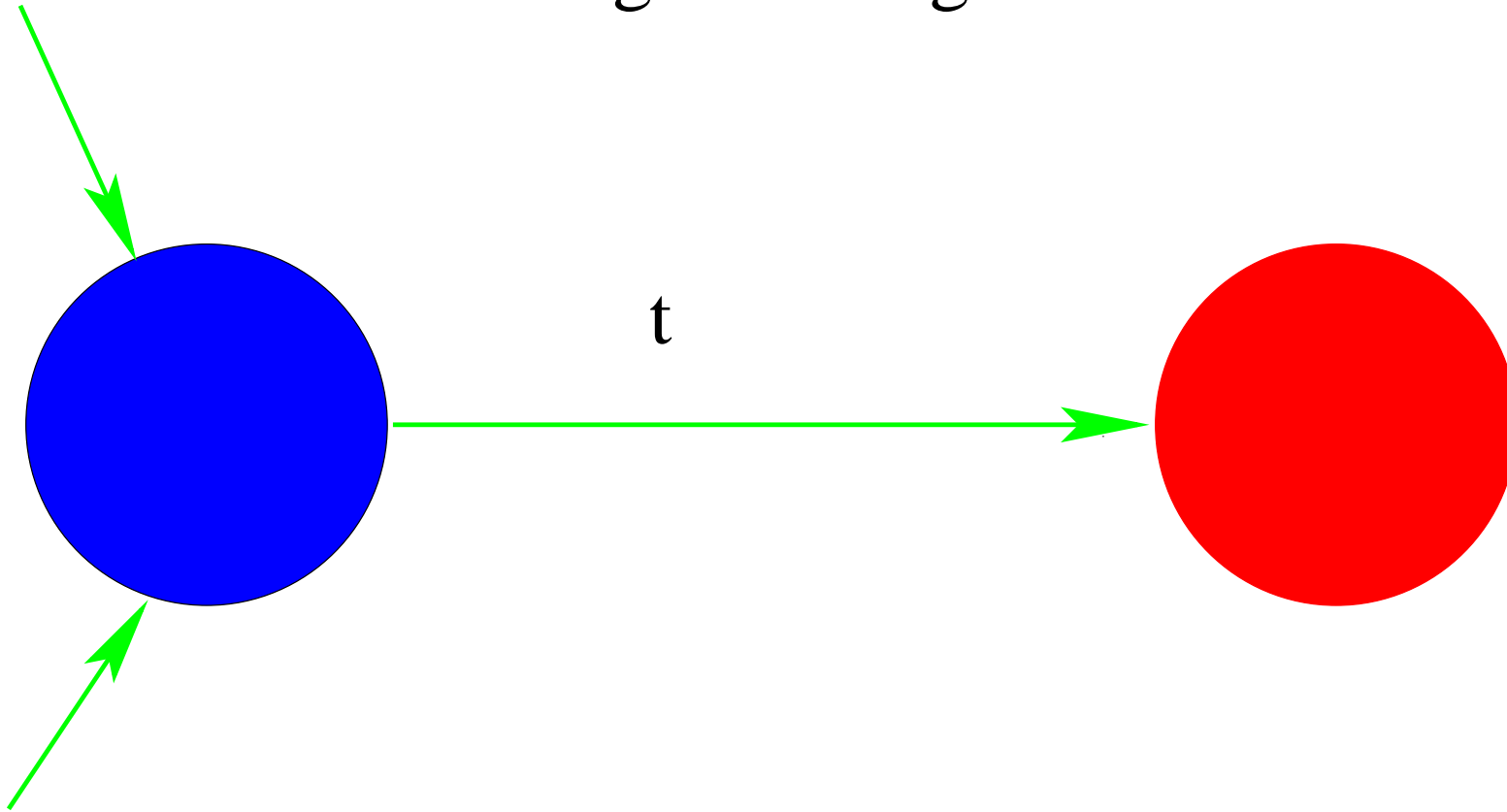




Sending a message



Receiving a message



Invariant (1)

$$m \in N \leftrightarrow N$$

$$t \subseteq m \subseteq g$$

Node x sends a **message** to node y

send_msg $\hat{=}$

any x, y **where**

$$x \mapsto y \in g \wedge$$

$$g[\{x\}] = t^{-1}[\{x\}] \cup \{y\} \wedge$$

$$y \mapsto x \notin t \wedge$$

$$x \notin \text{dom}(m)$$

then

$$m := m \cup \{x \mapsto y\}$$

end

Node y receives the **message** from node x Node y
accepts the **submission** from node x

```
progress  $\hat{=}$   
  any  $x, y$  where  
     $x \mapsto y \in m - t \wedge$   
     $y \notin \text{dom}(m)$   
  then  
     $t := t \cup \{x \mapsto y\}$   
  end
```

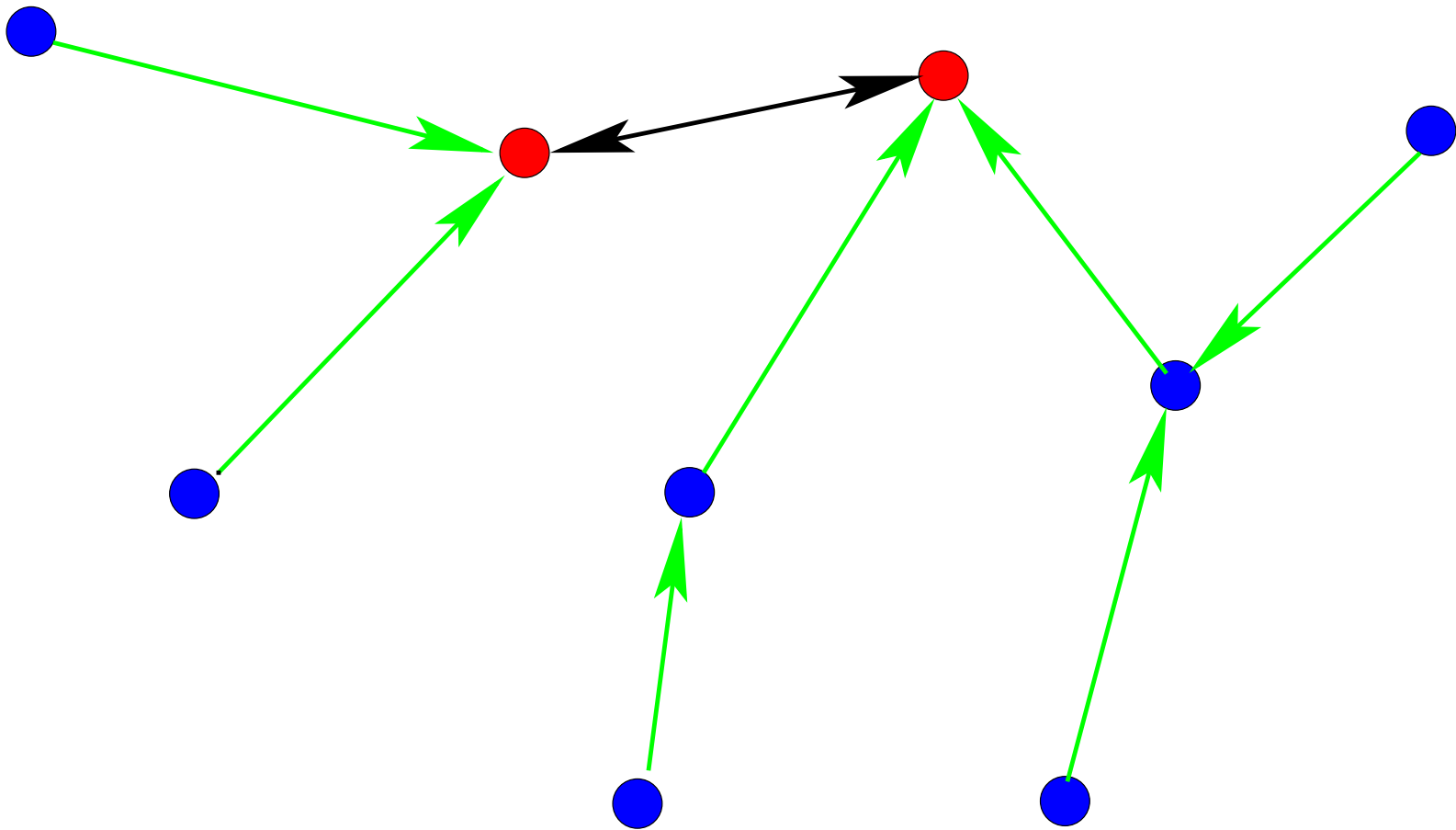
Invariant (2)

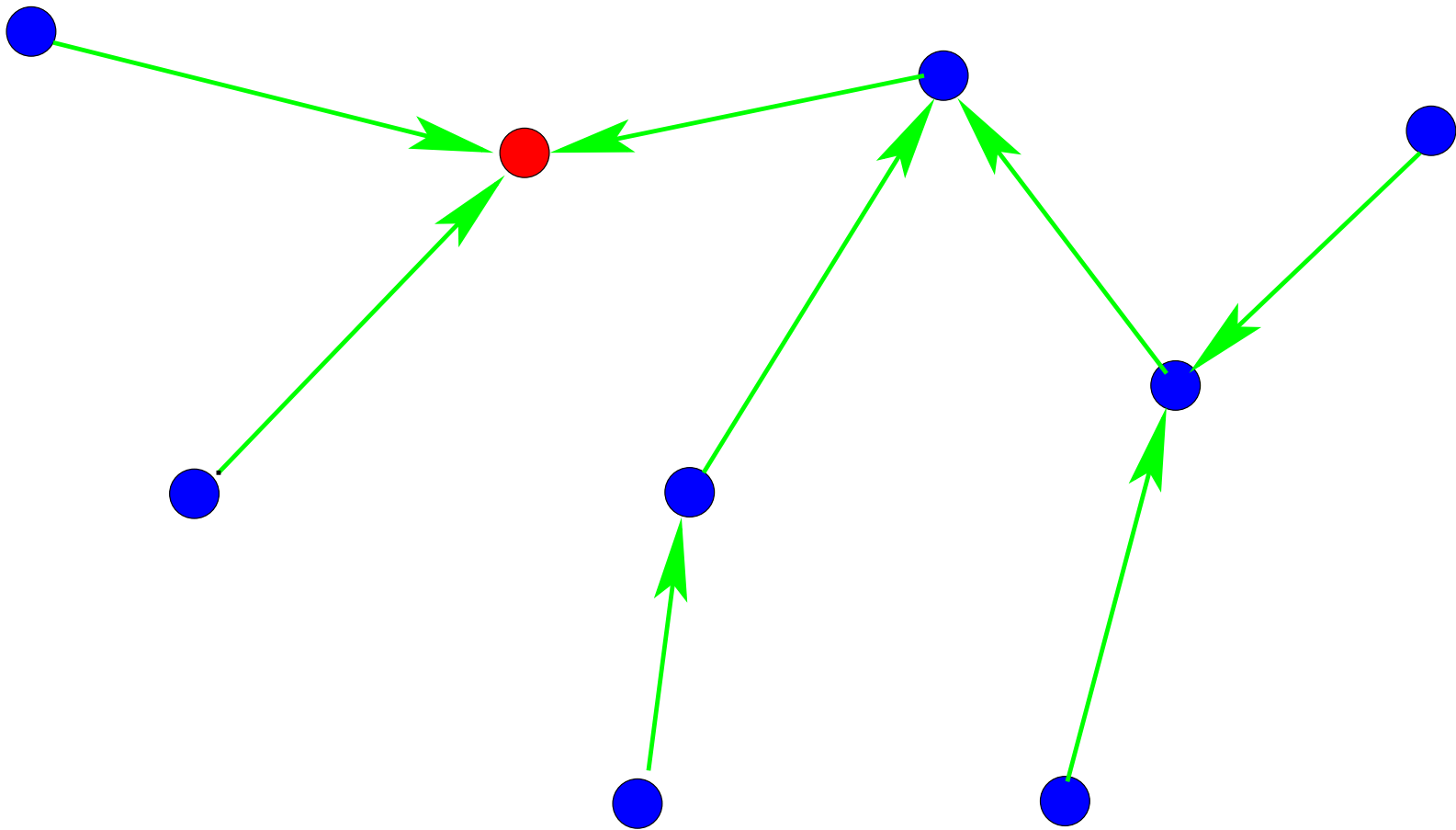
$$\forall (x, y) \cdot \left(\begin{array}{l} x \mapsto y \in m \\ \Rightarrow \\ g[\{x\}] = t^{-1}[\{x\}] \cup \{y\} \end{array} \right)$$

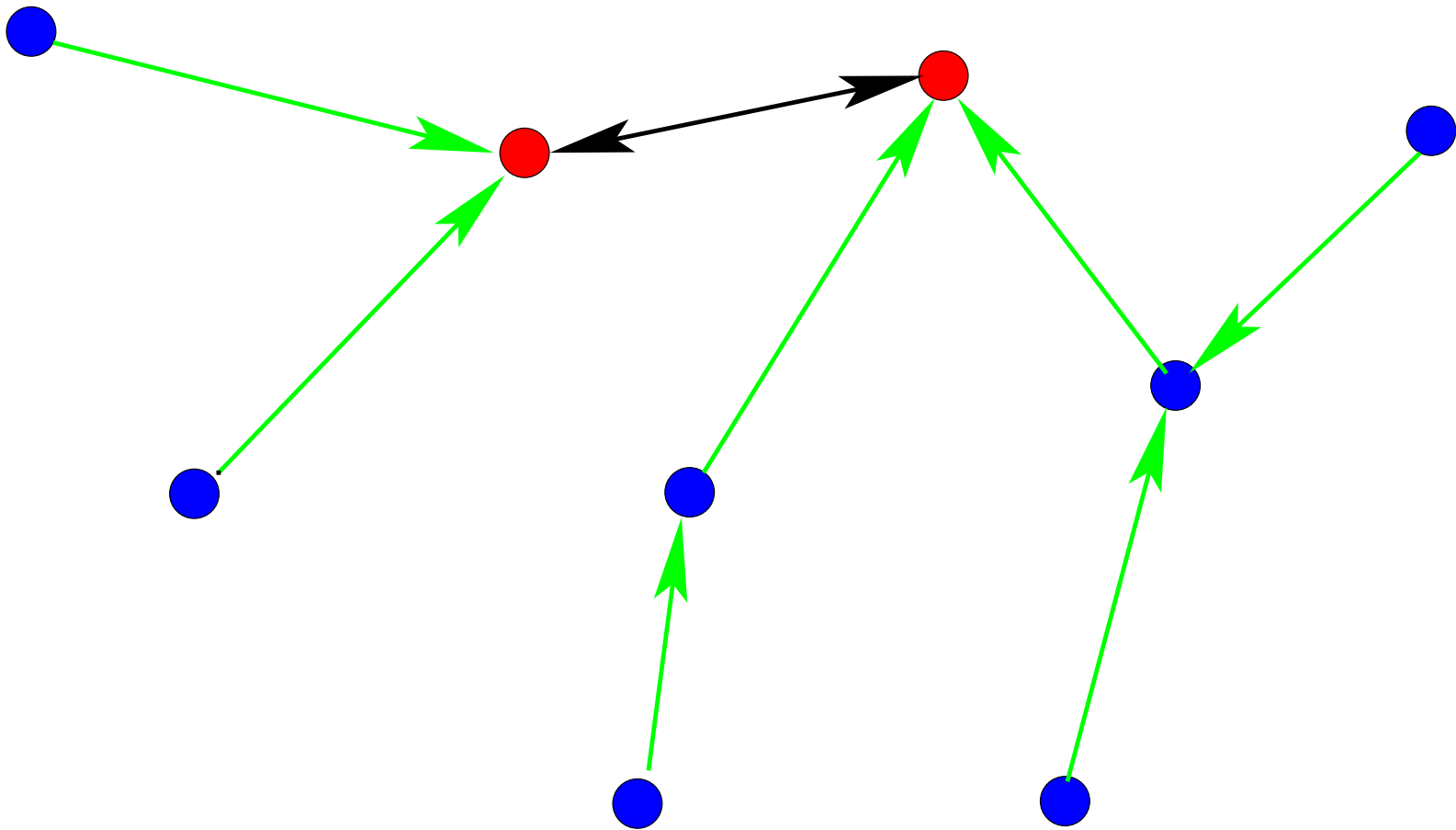
$$\forall (x, y) \cdot \left(\begin{array}{l} x \mapsto y \in m - t \quad \wedge \\ y \in \text{dom}(m) \\ \Rightarrow \\ y \mapsto x \in m - t \quad \wedge \end{array} \right)$$

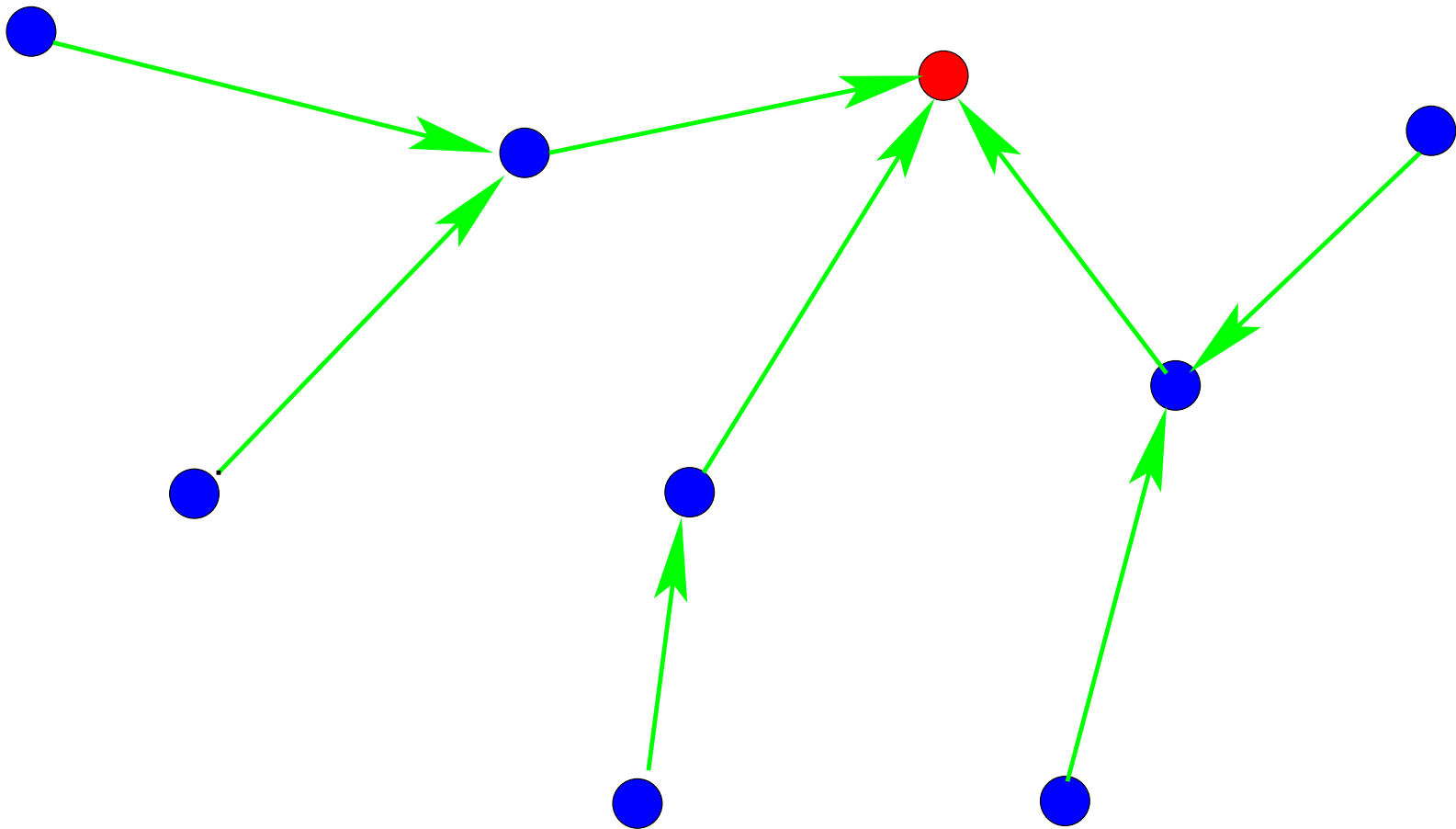
Second Refinement: The problem of contention

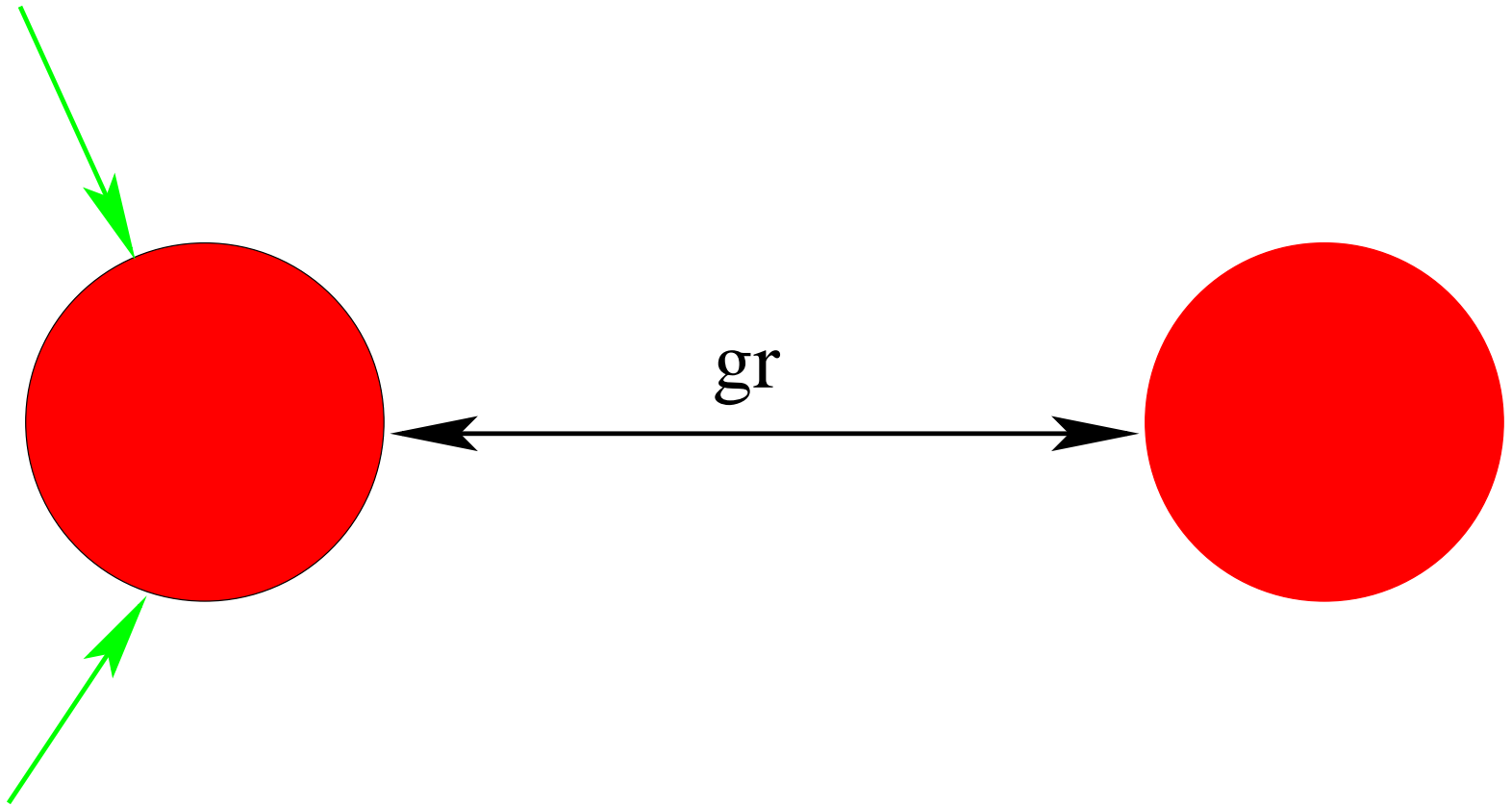
- Explaining the **problem**
- Proposing a **partial** solution
- Towards a **better** treatment
- Back to the **local** animation



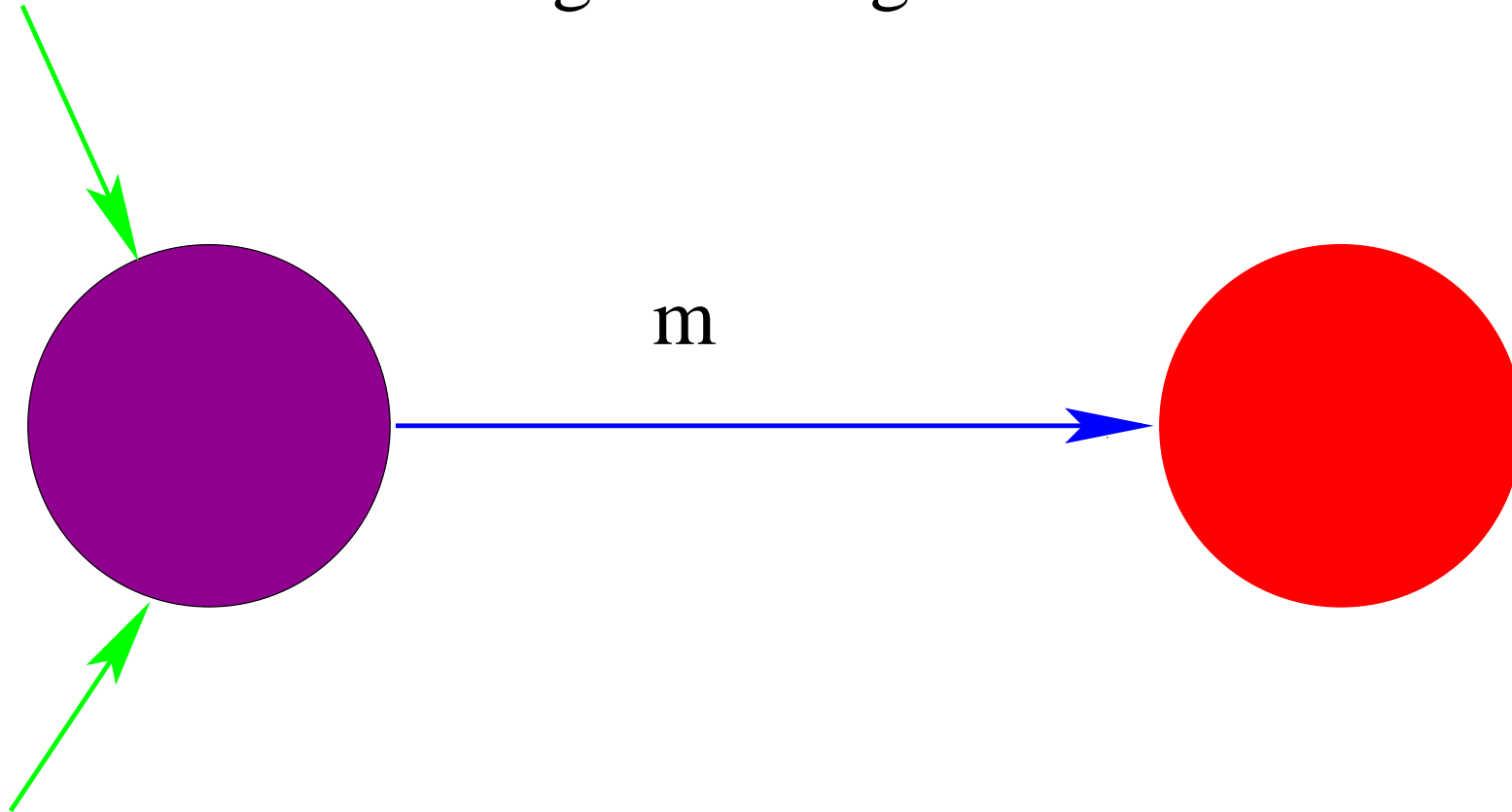




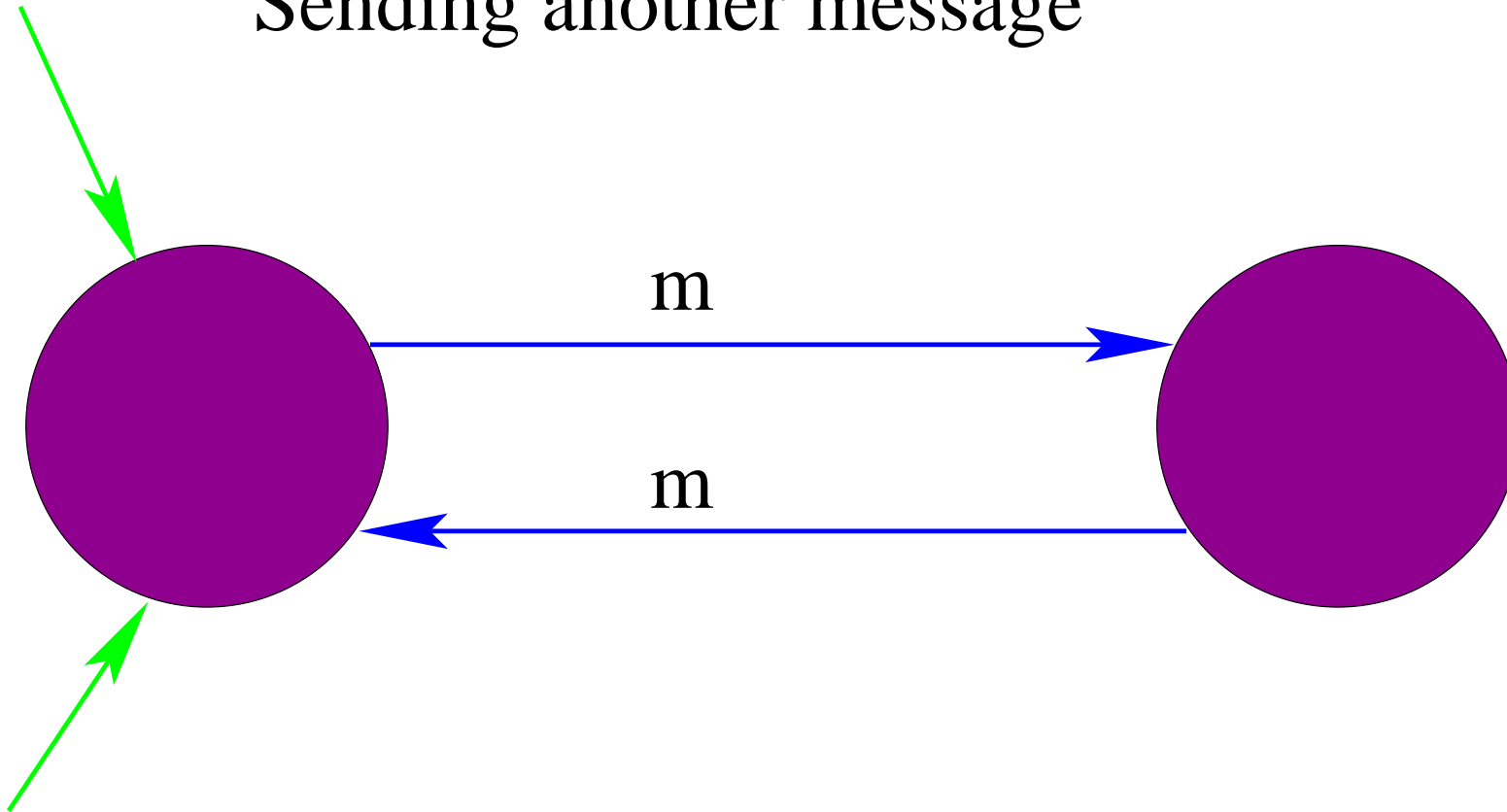




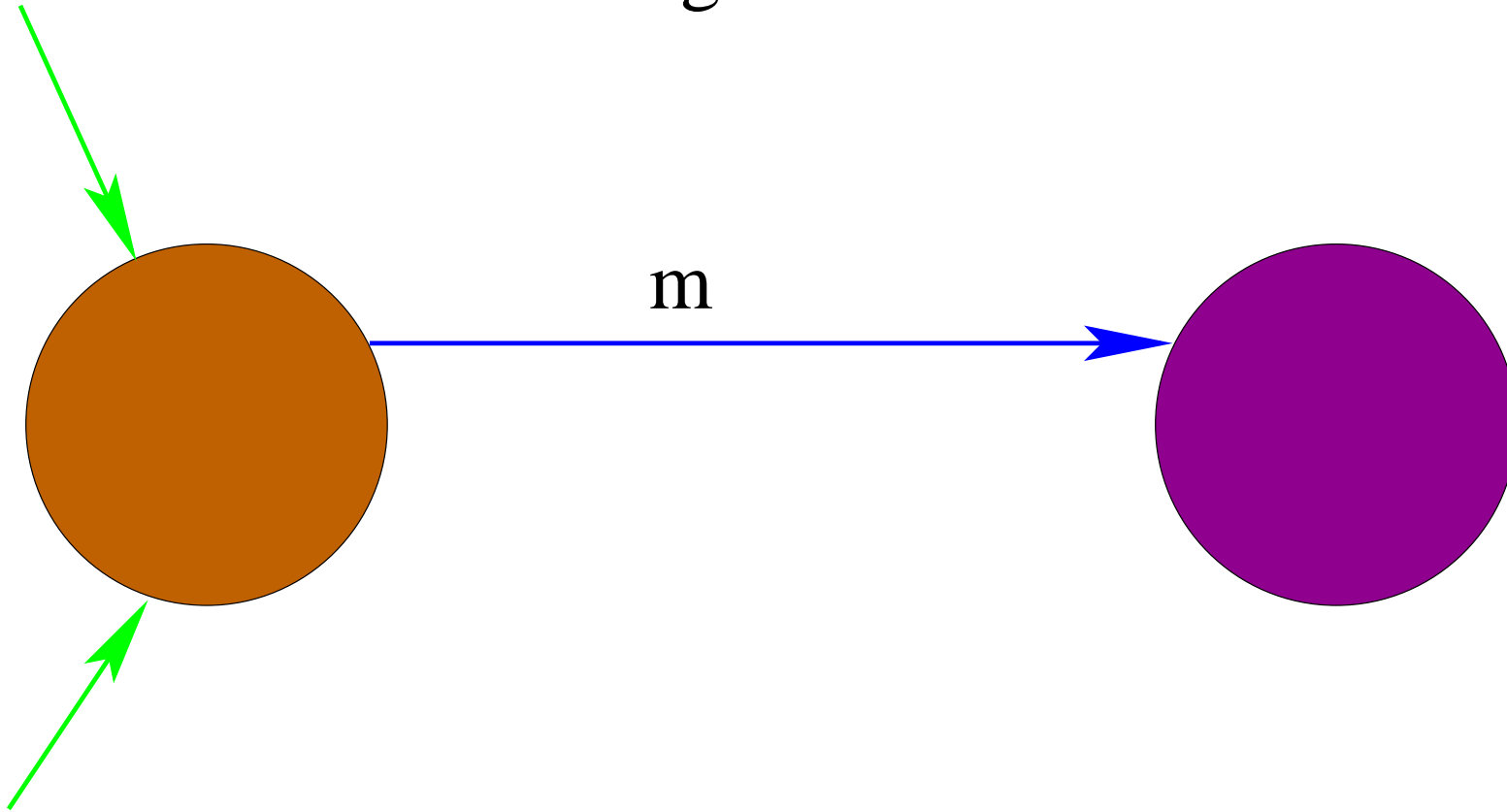
Sending a message



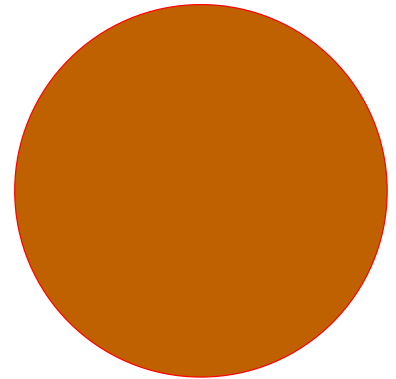
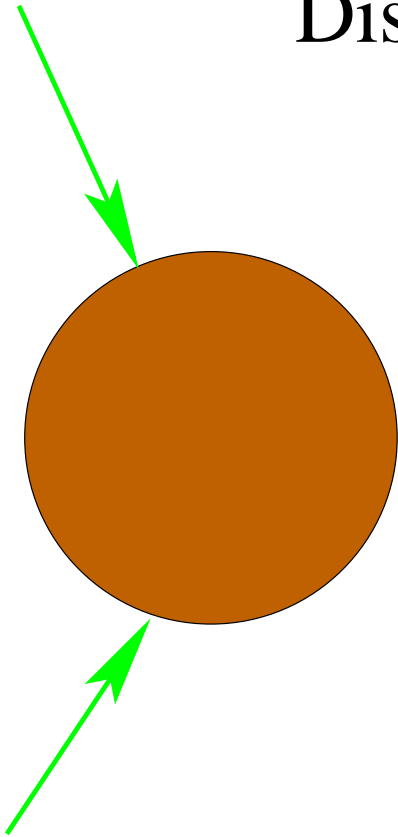
Sending another message



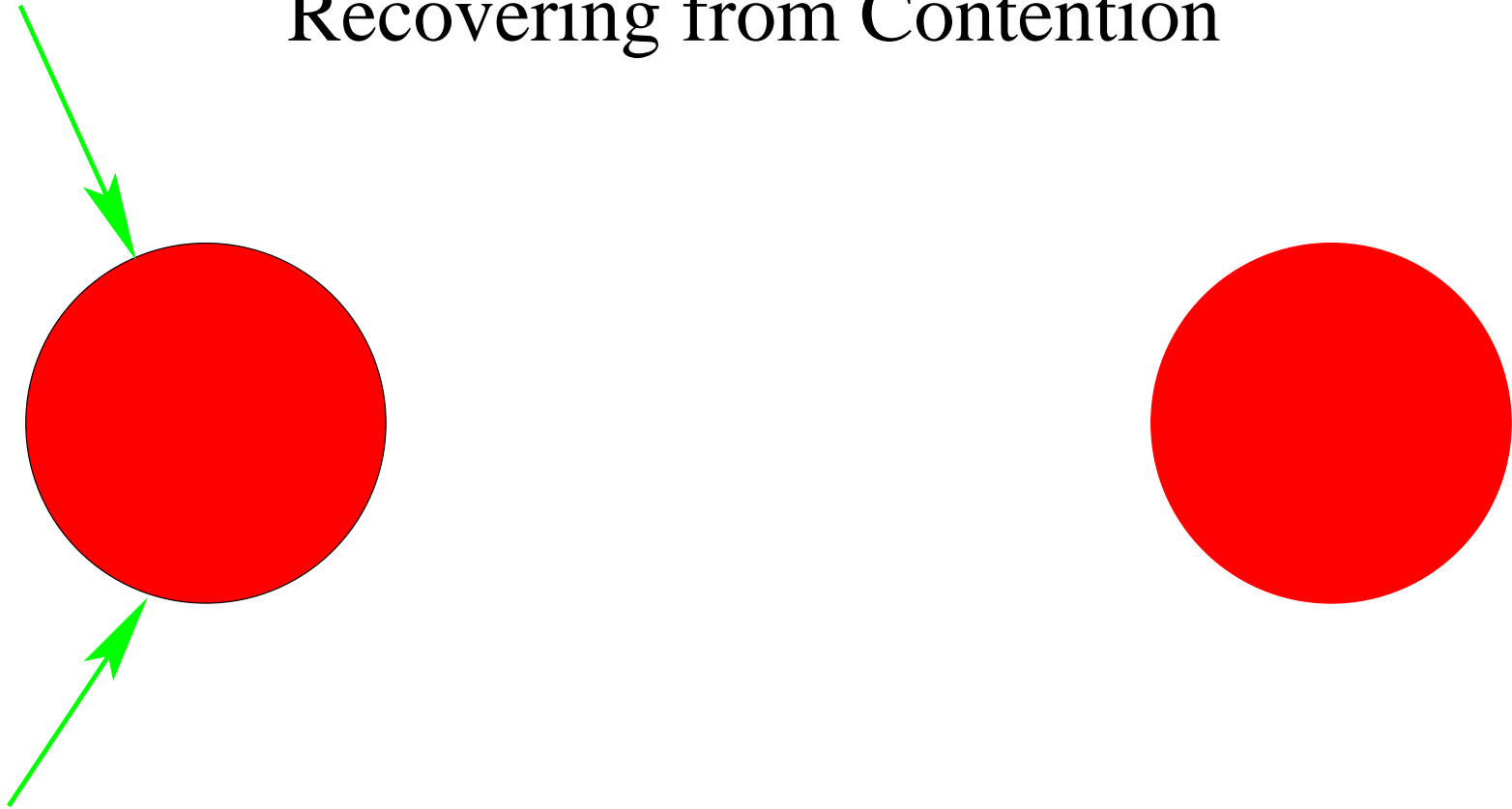
Discovering Contention



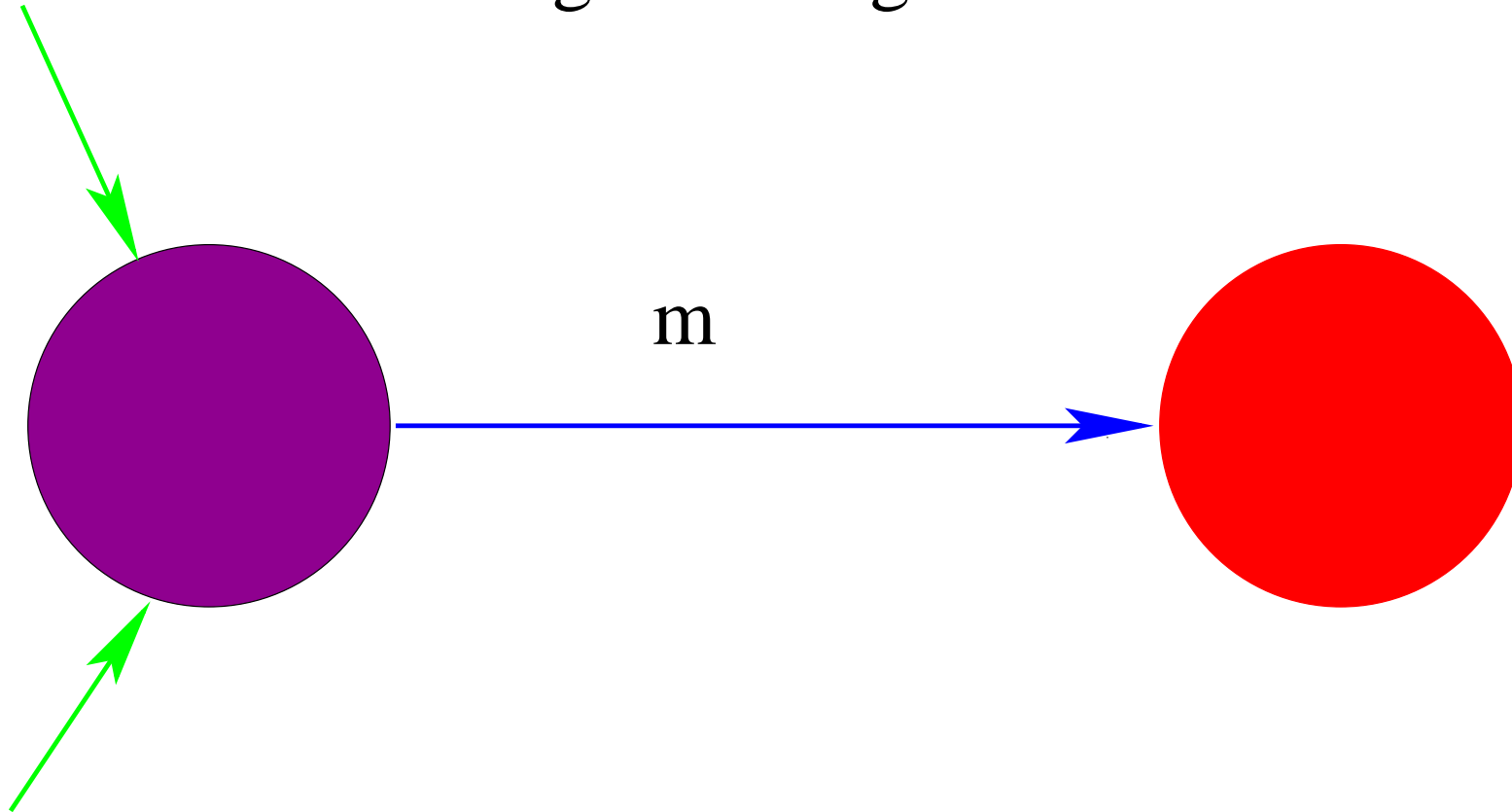
Discovering Contention



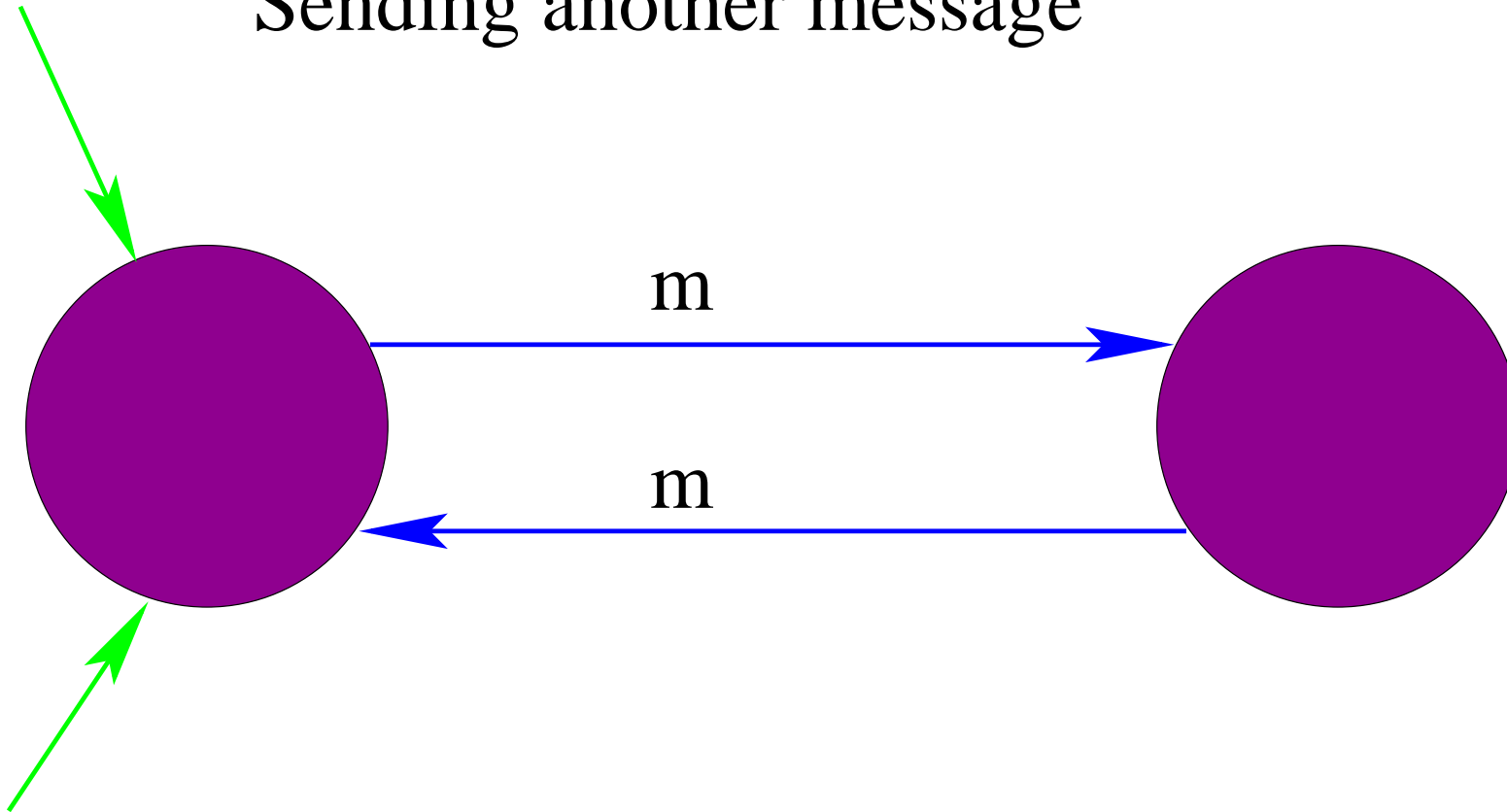
Recovering from Contention



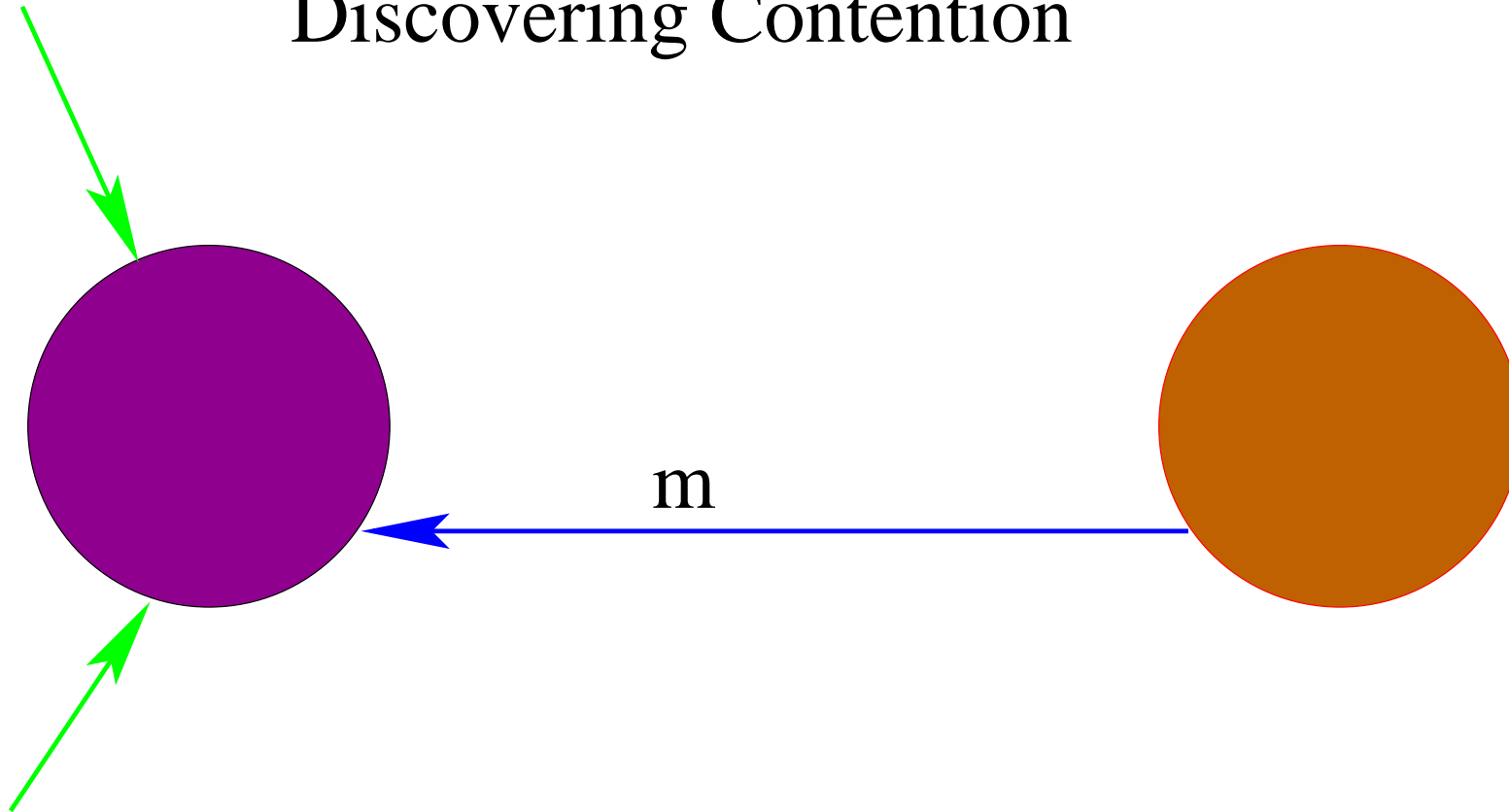
Sending a message



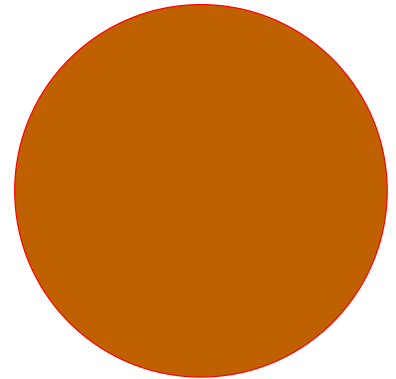
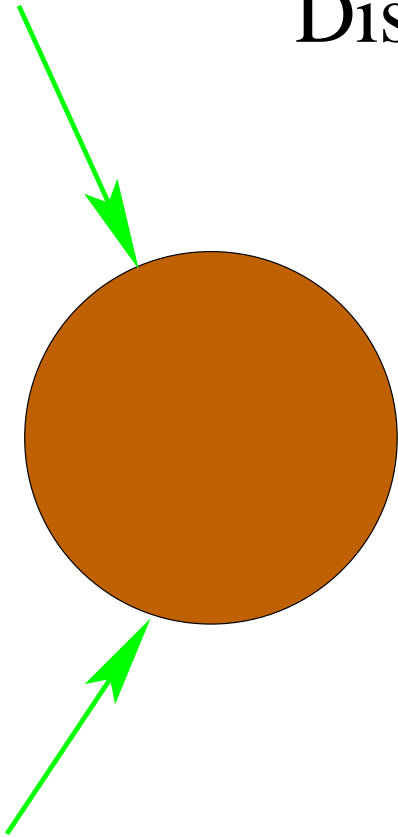
Sending another message



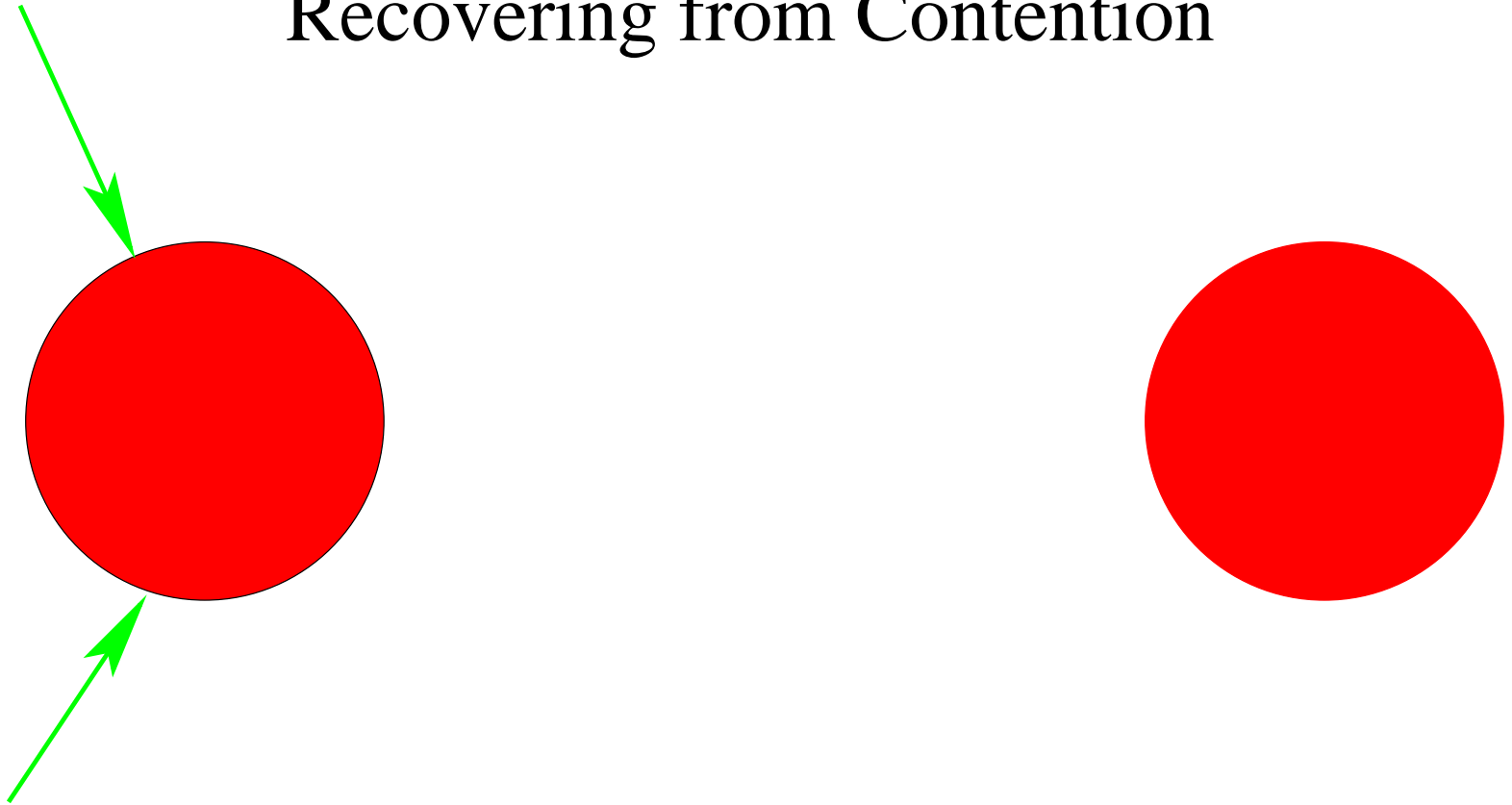
Discovering Contention



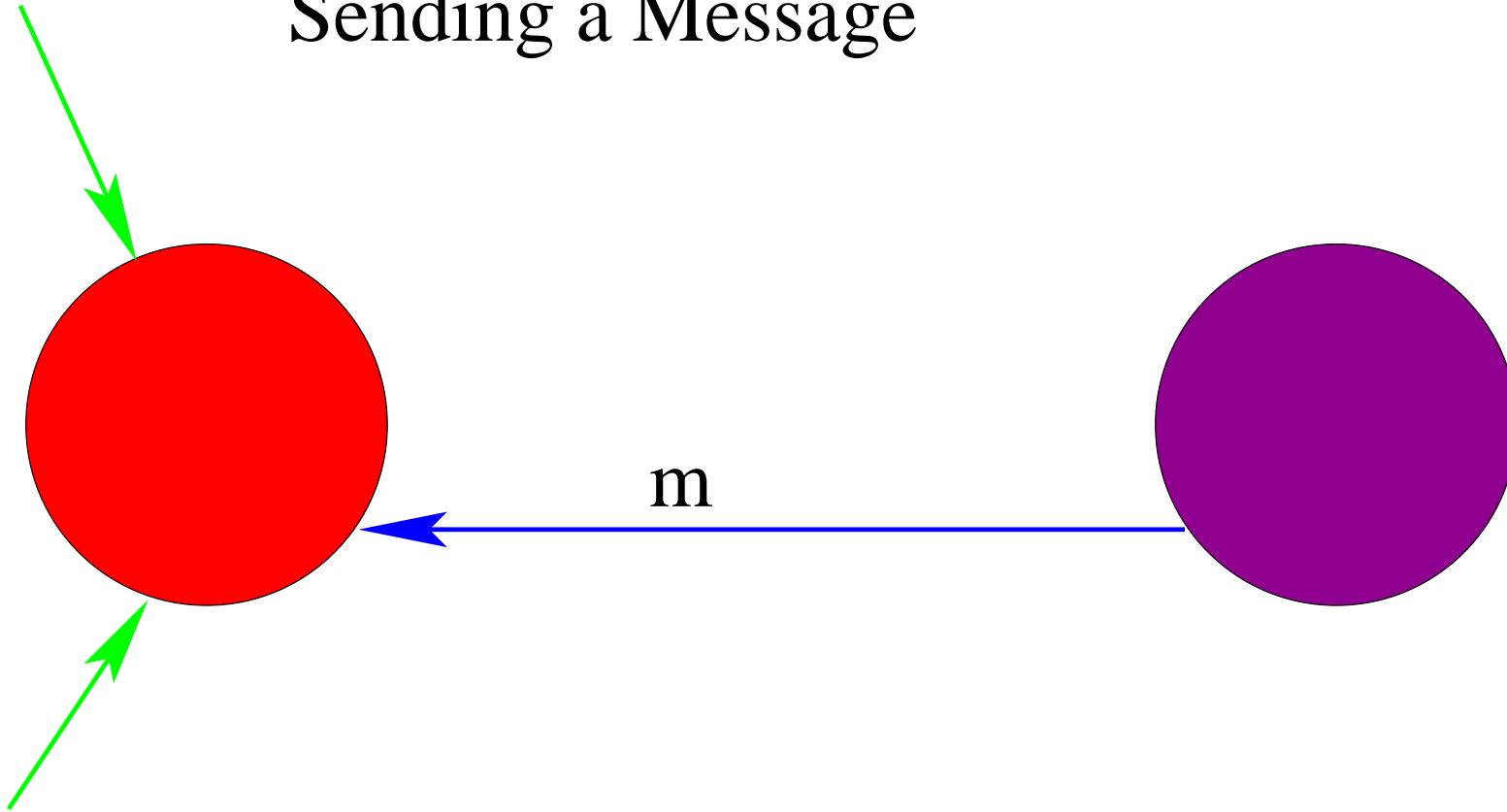
Discovering Contention



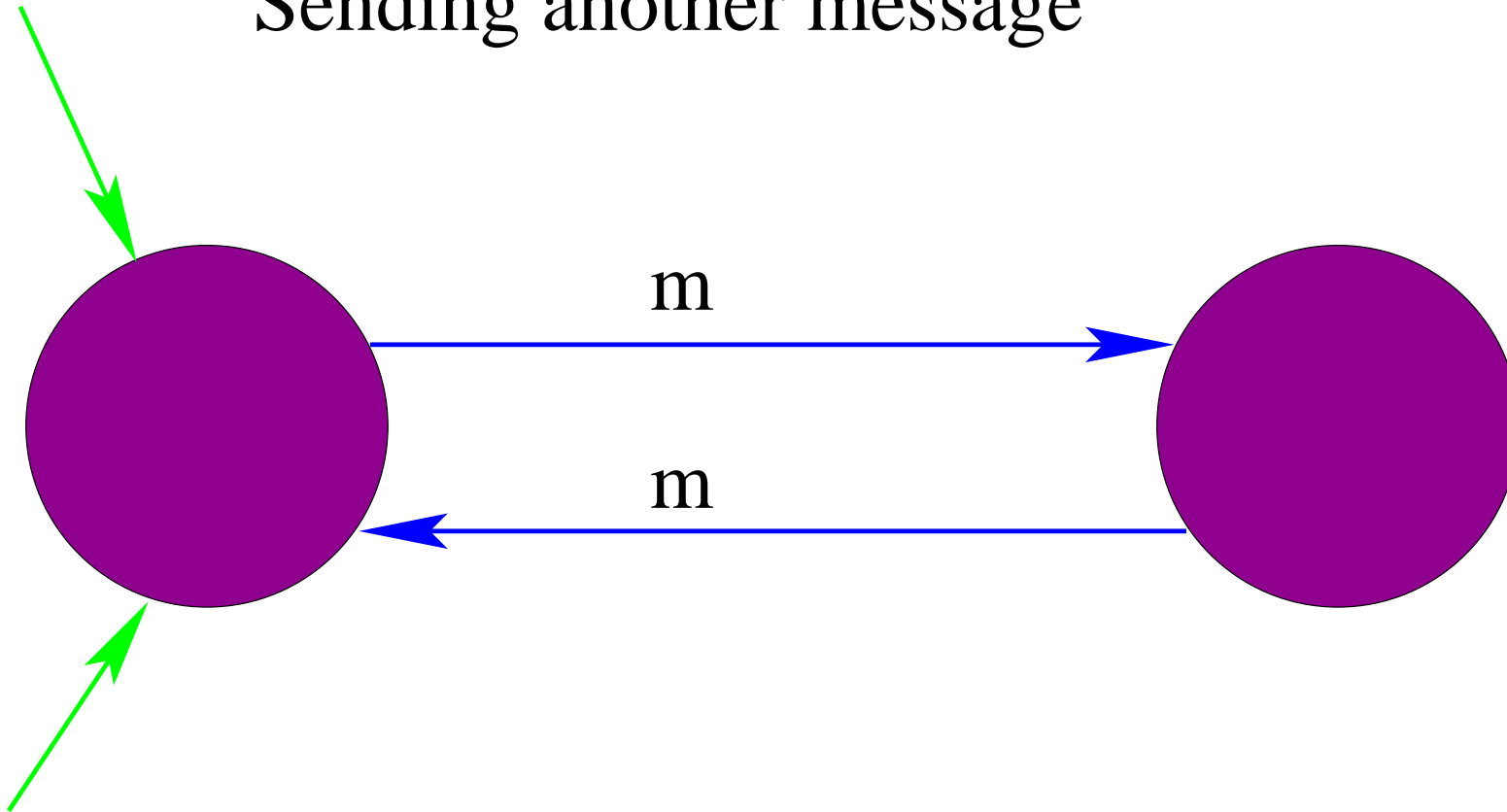
Recovering from Contention



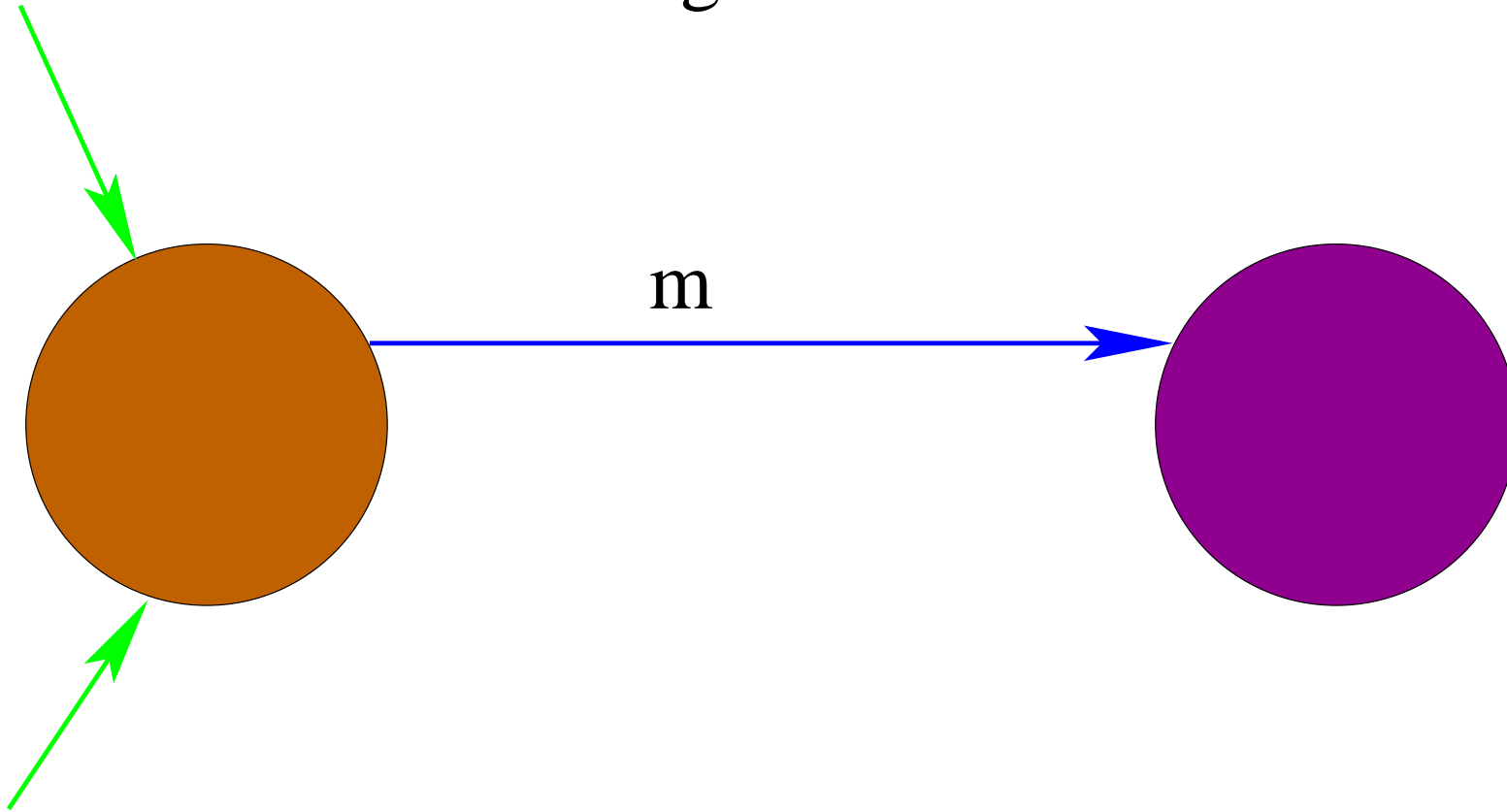
Sending a Message



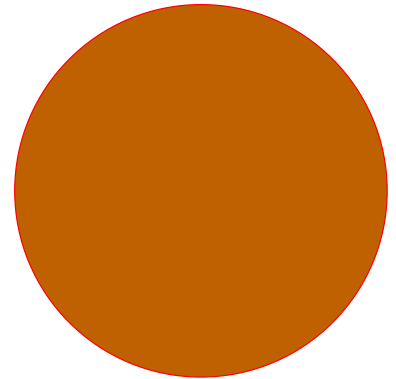
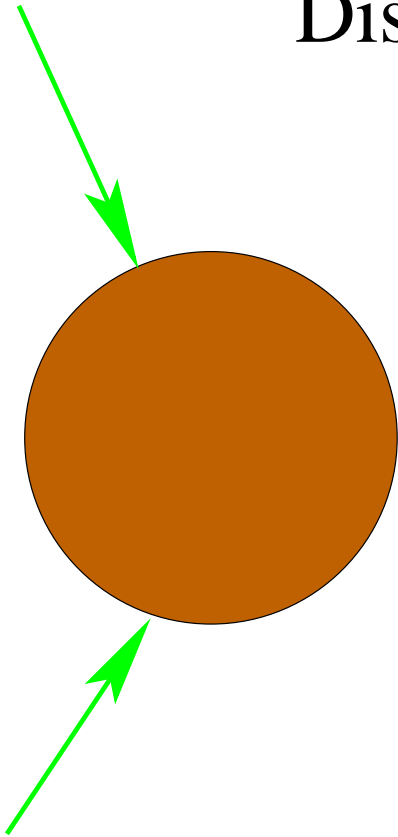
Sending another message

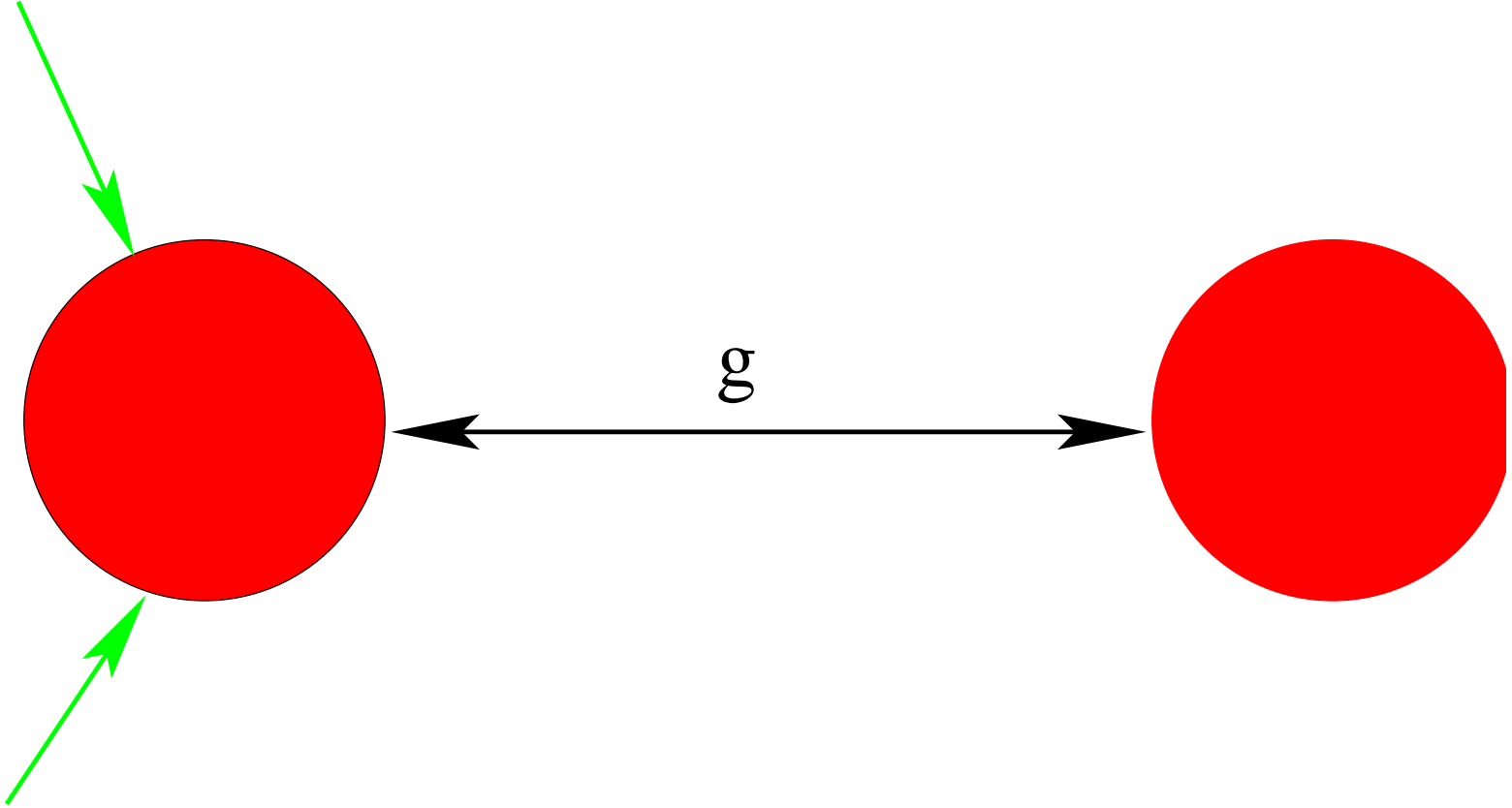


Discovering Contention

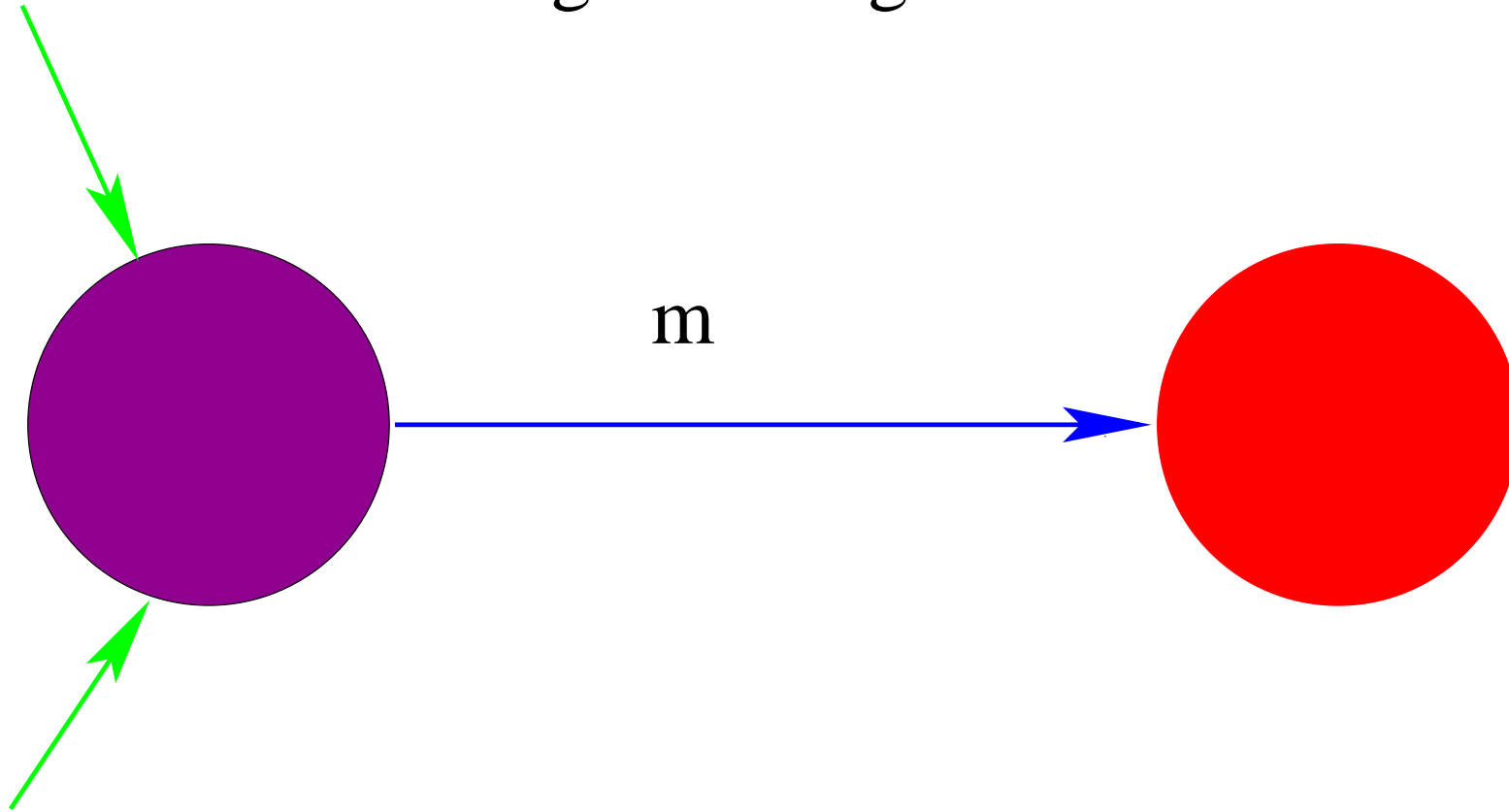


Discovering Contention

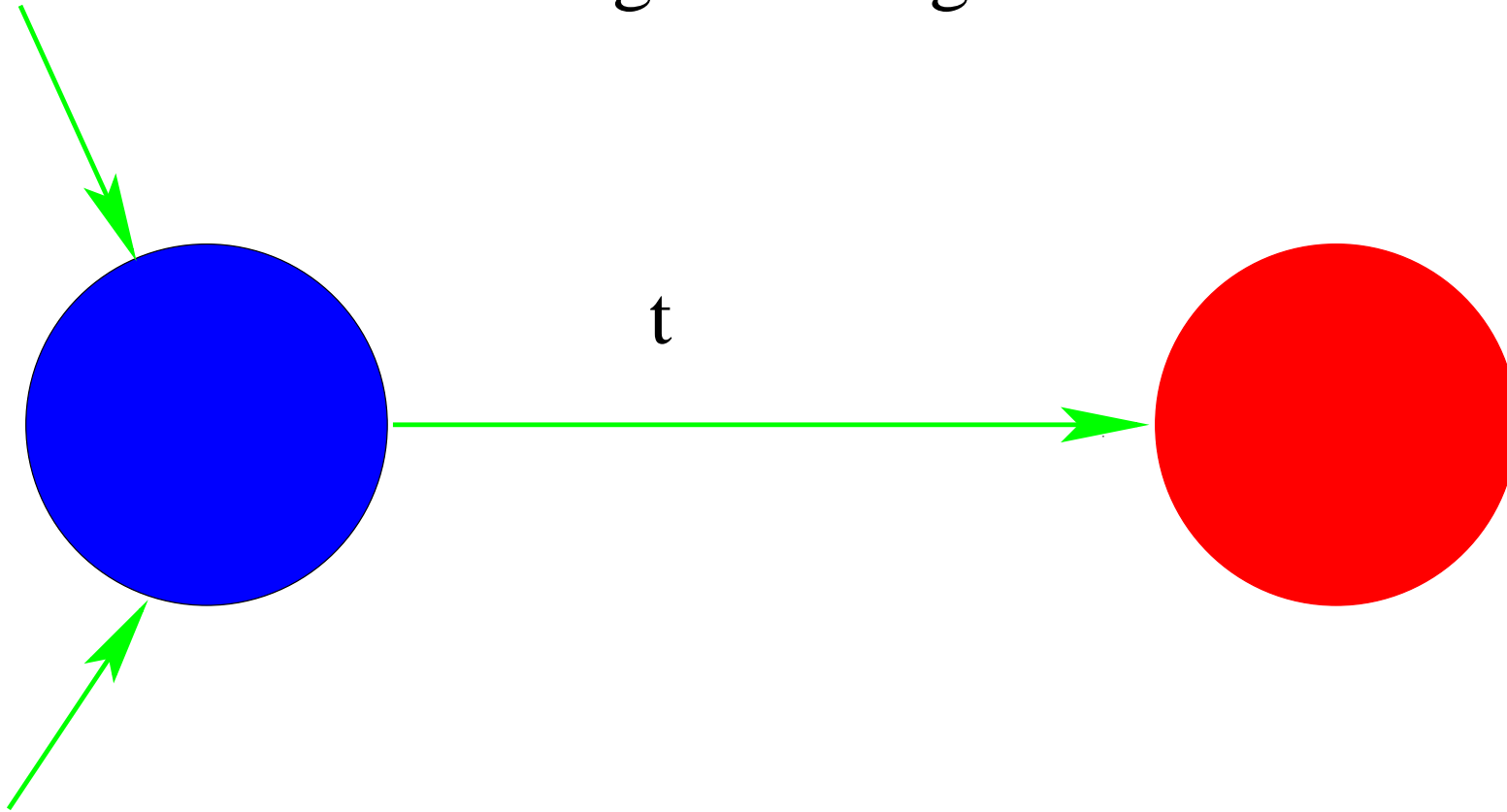




Sending a message



Receiving a message



Discovering the Contention (1)

- Node y **discovers the contention** with node x because:
 - It has **sent a message** to node x
 - It has **not yet received** a response from node x
 - It **receives instead** a message from node x

Discovering the Contention (2)

- Node x also **discovers the contention** with node y
- Assumption: The time **between both discoveries**
IS SUPPOSED TO BE BOUNDED
BY τ ms
- The time τ is the **maximum transmission time**
between 2 connected nodes

A Partial Solution

- Each node **waits for τ ms** after its own discovery
- After this, each node thus **knows** that the other has also discovered the contention
- Each node then **retries immediately**
- **PROBLEM:** This may continue **for ever**

A Better Solution (1)

- Each node **waits for τ ms** after its own discovery
- Each node then chooses **with equal probability**:
 - either to wait for a **short** delay
 - or to wait for a **large** delay
- Each node then **retries**

A Better Solution (2)

- **Question:** Does this solves the problem ?
- Are we sure to **eventually** have one node winning ?
- **Answer:** Yes, see **Caroll Morgan's works**

Node y discovers a contention with node x

progress $\hat{=}$
any x, y **where**
 $x \mapsto y \in m - t \wedge$
 $y \notin \text{dom}(m)$
then
 $t := t \cup \{x \mapsto y\}$
end

contention $\hat{=}$
any x, y **where**
 $x \mapsto y \in m - t \wedge$
 $y \in \text{dom}(m)$
then
 $c := c \cup \{x \mapsto y\}$
end

- Introducing a dummy contention channel: c

$$c \in N \leftrightarrow N$$

$$c \subseteq m$$

$$t \cap c = \emptyset$$

Invariant (3)

$$\forall (x, y) \cdot \left(\begin{array}{l} x \mapsto y \in m - t \wedge \\ y \notin \text{dom}(m) \\ \Rightarrow \\ x \mapsto y \notin c \end{array} \right)$$

$$\text{ran}(c) \cap \text{dom}(m) = \emptyset$$

$$(c \neq \emptyset \Rightarrow \exists(x, y) \cdot (x \mapsto y \in m - t \wedge y \in \text{dom}(m)))$$

Assertions

$$m \cap id(N) = \emptyset$$

$$c \subseteq (N - \text{dom}(t)) \times (N - \text{dom}(t))$$

Solving the contention (simulating the τ delay)

solve_contention $\hat{=}$

any x, y **where**

$$x \in N \wedge y \in N \wedge$$

$$c = \{x \mapsto y, y \mapsto x\}$$

then

$$m := m - c \quad ||$$

$$c := \emptyset$$

end

Summary of Second Refinement

- 38 proofs
- Among which 0 were interactive (only one click p0 or p1)

Third Refinement: Removing m

- In the abstraction the message is not removed when it is received
- M are the concrete messages which travel through the graph g
- bm is a boolean to know if a node has sent a message

Invariant

$$M \subseteq N \times N$$

$$m = M \cup t \cup c$$

$$M \cap t = \emptyset$$

$$M \cap c = \emptyset$$

$$bm = \text{dom}(m)$$

Node x sends a **message** to node y

send_msg $\hat{=}$

any x, y **where**

$$x \mapsto y \in g \wedge$$

$$g[\{x\}] = t^{-1}[\{x\}] \cup \{y\} \wedge$$

$$y \mapsto x \notin t \wedge$$

$$x \notin bm$$

then

$$M := M \cup \{x \mapsto y\} ||$$

$$bm := bm \cup \{x\}$$

end

progress $\hat{=}$

any x, y **where**

$x \mapsto y \in M \wedge$

$y \notin bm$

then

$t := t \cup \{x \mapsto y\} ||$

$M := M - \{x \mapsto y\}$

end

contention $\hat{=}$

any x, y **where**

$x \mapsto y \in M \wedge$

$y \in bm$

then

$c := c \cup \{x \mapsto y\}$

$M := M - \{x \mapsto y\}$

end

solve_contention $\hat{=}$

any x, y **where**

$$x \in N \wedge y \in N \wedge$$

$$c = \{x \mapsto y, y \mapsto x\}$$

then

$$bm := bm - \{x, y\} \quad ||$$

$$c := \emptyset$$

end

Summary of Third Refinement

- 23 proofs
- Among which 2 were interactive (not only one click p0 or p1)

Fourth Refinement: Removing c

- introduce the set C of nodes in contention (semi-localisation)
- $C = \text{ran}(c)$
- It's time to prove that there are only two nodes when the contention occurs

contention $\hat{=}$

any x, y **where**

$x \mapsto y \in M \wedge$

$y \in bm$

then

$C := C \cup \{y\}$

$M := M - \{x \mapsto y\}$

end

solve_contention $\hat{=}$

any x, y **where**

$$x \in N \wedge y \in N \wedge$$

$$x \neq y \wedge$$

$$C = \{x, y\}$$

then

$$bm := bm - \{x, y\} \quad ||$$

$$C := \emptyset$$

end

$$\forall (x, y) \cdot \left(\begin{array}{l} x \mapsto y \in g \wedge \\ x \notin \text{dom}(t) \wedge \\ y \notin \text{dom}(t) \wedge \\ g[\{x\}] = t^{-1}[\{x\}] \cup \{y\} \wedge \\ g[\{y\}] = t^{-1}[\{y\}] \cup \{x\} \\ \Rightarrow \\ \text{dom}(t) = N - \{x, y\} \end{array} \right)$$

$$\forall (x, y) \cdot \left(\begin{array}{l} x \in C \wedge y \in C \wedge x \neq y \\ \Rightarrow \\ c = \{x \mapsto y, y \mapsto x\} \end{array} \right)$$

Summary of Fourth Refinement

- 8 proofs
- Among which 3 were interactive (not only one click)

Fith Refinement: Localization

- The representation of the **graph g** is **modified**
- The representation of the **tree t** is **modified**
- Other data structures are **localized**

Localization (1)

The graph g and the tree t are now **localized**

$$n \in N \rightarrow \mathbb{P}(N)$$

$$\forall x \cdot (x \in N \Rightarrow n(x) = g[\{x\}])$$

$$s \in N \rightarrow \mathbb{P}(N)$$

$$\forall x \cdot (x \in N \Rightarrow s(x) = t^{-1}[\{x\}])$$

- Node x is elected the leader

elect $\hat{=}$

any x **where**

$$x \in N \wedge$$

$$n(x) = s(x)$$

then

$$l := x$$

end

- Node x sends a message to node y (y is unique)

send_msg $\hat{=}$

any x, y **where**

$$x \in N - bm \wedge$$

$$y \in N - s(x) \wedge$$

$$n(x) = s(x) \cup \{y\}$$

then

$$M := M \cup \{x \mapsto y\} \quad ||$$

$$bm := bm \cup \{x\}$$

end

- Node y receives the message from node x

progress $\hat{=}$

any x, y **where**

$x \mapsto y \in M \wedge$

$y \notin bm$

then

$s(y) := s(y) \cup \{x\} \quad ||$

$M := M \cup \{x \mapsto y\}$

end

Summary of Fith Refinement

- 11 proofs
- Among which 3 were interactive

Sixth Refinement: Preparing counters

- Replacing n and b by d :

$$\forall x \cdot (x \in N \Rightarrow d(x) = n(x) - s(x))$$

- guard of elect $d(x) = \emptyset$

- guard of send_msg $d(x) = \{y\}$

- Summary **9** proofs Among which **4 were interactive**

Seventh Refinement: counters

- Adding $r: \forall x \cdot (x \in N \Rightarrow r(x) = \text{card}(d(x)))$
- guard of elect $r(x) = 0$
- guard of send_msg $r(x) = 1$
- Summary **8** proofs Among which **4 were interactive**

Eighth Refinement: wires

- Replacing d by D : $D \in N \rightarrow (N \rightarrow \{low, high\})$
- $\forall x \cdot (x \in N \Rightarrow \text{dom}(D(x)) = n(x))$
- $\forall x \cdot (x \in N \Rightarrow d(x) = D(x)^{-1}[\{low\}])$
- Replacing bm by B : $B \in N \rightarrow \{low, high\}$ and $bm = B^{-1}[\{high\}]$
- Summary **20** proofs Among which **7 were interactive**

- Node x sends a message to node y (y is unique)

send_msg $\hat{=}$

any x, y **where**

$$x \in N \wedge$$

$$B(x) = low \wedge$$

$$y \in D(x)^{-1}[\{low\}] \wedge$$

$$r(x) = 1$$

then

$$M := M \cup \{x \mapsto y\} \quad ||$$

$$B(x) := high$$

end

- Node y receives the message from node x

progress $\hat{=}$

any x, y **where**

$$x \mapsto y \in M \wedge$$

$$B(y) = low$$

then

$$D(y)(x) := high \quad ||$$

$$r(y) := r(y) - 1 \quad ||$$

$$M := M - \{x \mapsto y\}$$

end

Main Summary

- 129 proofs
- Among which 26 were interactive

Conclusion: a Systematic Approach to Distribution

- Establishing the **mathematical framework**

Conclusion: a Systematic Approach to Distribution

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**

Conclusion: a Systematic Approach to Distribution

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**
- Resolving the same problem on a **step by step basis**

Conclusion: a Systematic Approach to Distribution

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**
- Resolving the same problem on a **step by step basis**
- Involving communication **by means of messages**

Conclusion: a Systematic Approach to Distribution

- Establishing the **mathematical framework**
- Resolving the mathematical problem in **one shot**
- Resolving the same problem on a **step by step basis**
- Involving communication **by means of messages**
- Towards the **localization of data structures**