Event-B Course

12. Synchronizing Processes on a Tree Network

Jean-Raymond Abrial

September-October-November 2011

Purpose of this Lecture

- Learning a few more modeling conventions

- Learning more about abstraction

- Learning how to formalize an interesting structure: a tree

- Study a more complicated problem in distributed computing

- Example studied in the following book:

  W.H.J. Feijen and A.J.M. van Gasteren.

  On a Method of Multi-programming Springer Verlag 1999.

Outline of Example

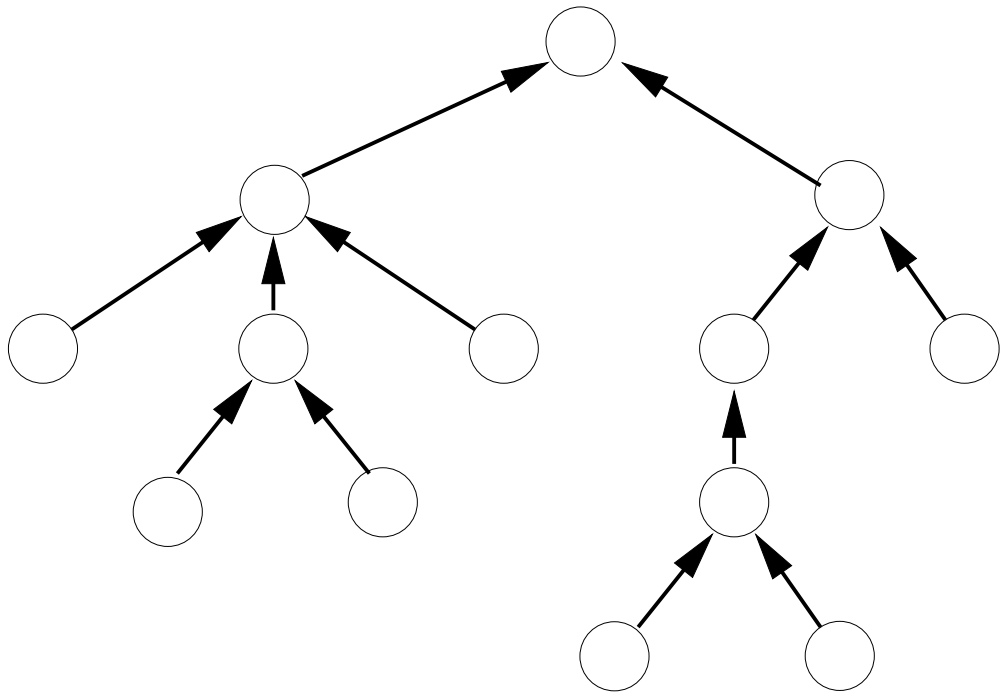
- Define the informal requirements

- Define the refinement strategy

- Construct the various more and more concrete models

Requirements (1)

We have a fixed set of processes forming a tree	ENV-1



Requirements (2)

- All processes are supposed to execute for ever the same code

- But processes must remain synchronized

- For this, we assign a counter to each process

Each process has a counter, which is a natural number	ENV-2

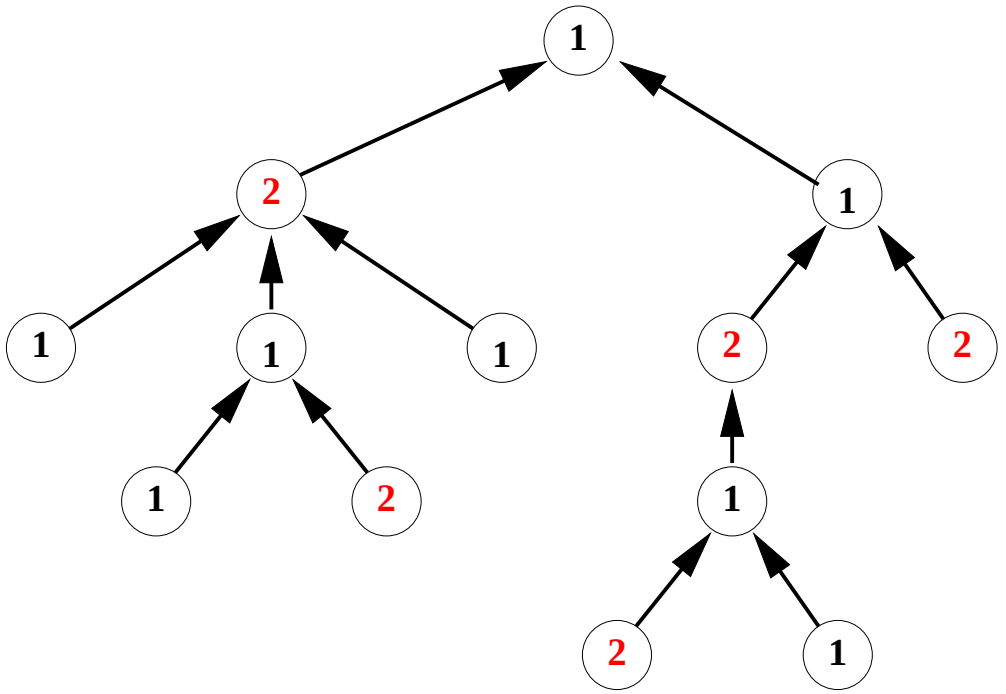
Being More Precise

- The counter of a process represents its “phase”

- The difference between any two counters is not greater than 1

- Each process is thus at most one phase ahead of the others

Requirements (3)



The difference between any two counters is at most equal to 1	FUN-1

Requirements (4)

- Reading the counters

Each process can read the counters of its immediate neighbors only	FUN-2

- Modifying the counters

The counter of a process can be modified by this process only	FUN-3

Refinement Strategy

- Construct an abstract initial model dealing with FUN-1 and FUN-3
- Improve the design to (partially) take care of FUN-2
- Improve the design to better take care of FUN-2
- Simplify the final design to obtain an efficient implementation

The difference between any two counters is at most equal to 1	FUN-1
Each process can read the counters of its immediate neighbors only	FUN-2
The counter of a process can be modified by this process only	FUN-3

Initial Model: the State

- We simplify the situation: we forget about the tree

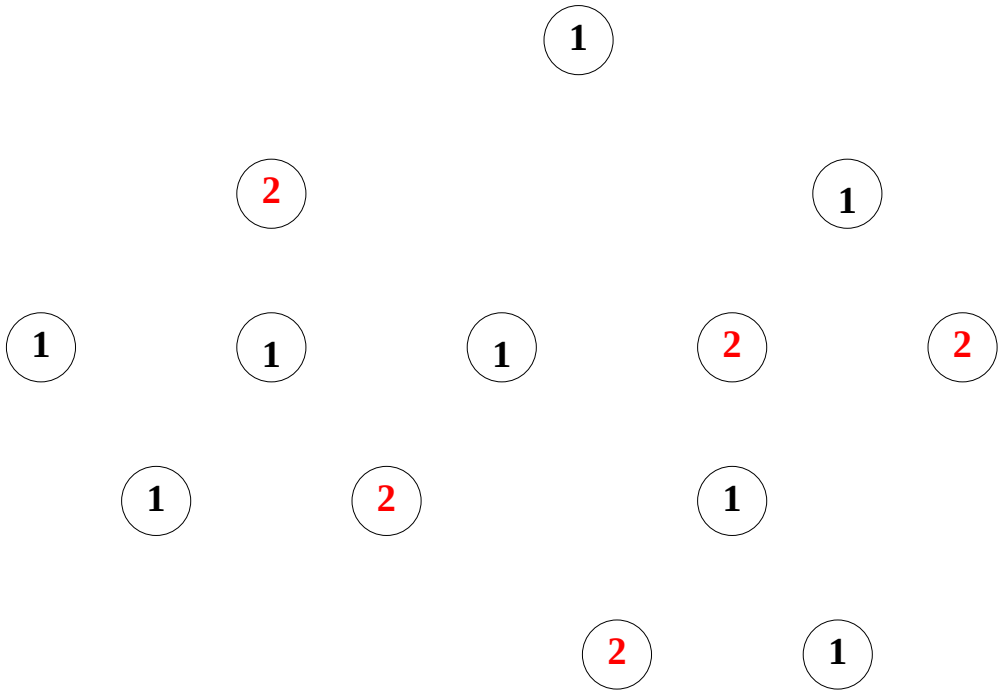
- We just define the counters and express the main property: FUN-1

The difference between any two counters is at most equal to 1	FUN-1

- The initial model is always far more abstract than the final system

- Other requirements are probably not fulfilled

Abstract Situation



The difference between any two counters is at most equal to 1	FUN-1

Initial Model: the State

set: P

axm0_1: $\mathrm{finite}(P)$

variable: c

inv0_1: $c \in P \to \mathbb{N}$

inv0_2: $\forall x, y \cdot c(x) \leq c(y) + 1$

We have:

$$-1 \leq c(x) - c(y) \leq 1$$

$$-1 \leq c(y) - c(x) \leq 1$$

that is

$$|c(x) - c(y)| \leq 1$$

Initial Model: the Events

init
$c := P \times \{0\}$

ascending
any n **where**
$n \in P$
$\forall m \cdot c(n) \leq c(m)$
then
$c(n) := c(n) + 1$
end

- A process counter is incremented only when $\leq$ to all other counters

- Notice the non-determinacy

Proof of inv0_2 Preservation by Event ascending

$c \in P \to \mathbb{N}$	**inv0_1**
$\forall x, y \cdot c(x) \leq c(y) + 1$	**inv0_2**
$n \in P$	Guards of event
$\forall m \cdot c(n) \leq c(m)$	ascending

$\vdash$

$$\forall x, y \cdot c \lhd\!\!+ \{n \mapsto c(n) + 1\})(x) \leq (c \lhd\!\!+ \{n \mapsto c(n) + 1\})(y) + 1$$

$\Uparrow$

Modified invariant **inv0_2**

Proof (cont'd)

$$
\begin{array}{l}
c \in P \to \mathbb{N} \\
\forall x, y \cdot c(x) \leq c(y) + 1 \\
n \in P \\
\forall m \cdot c(n) \leq c(m) \\
\vdash \\
(c \lhd \{n \mapsto c(n) + 1\})(x) \leq (c \lhd \{n \mapsto c(n) + 1\})(y) + 1
\end{array}
$$

- We perform then an easy proof by cases: $\begin{cases} x = n, y = n \\ x \neq n, y = n \\ x = n, y \neq n \\ x \neq n, y \neq n \end{cases}$

Other Verication Conditions to be Proved

- Initialisation and invariant establishment

- Liveness: a forgotten requirement

Once started, the system must work for ever	FUN-4

Problem with the Current Event

ascending
  any n **where**
    $n \in P$
    $\forall m \cdot c(n) \leq c(m)$
  then
    $c(n) := c(n) + 1$
  end

- Requirement FUN-2 is not fulfilled:

Each node can read the counters of its immediate neighbors only	FUN-2

First Refinement: Solving the problem (partially only)

- We introduce a special process r

- We suppose that the counter of r is always minimal

$$\forall m \cdot c(r) \leq c(m)$$

- This is a new invariant (for the moment)

First Refinement: Proposal for the Event Refinement

- We simplify the guard

(abstract-)ascending
any n **where**
$n \in P$
$\forall m \cdot c(n) \leq c(m)$
then
$c(n) := c(n) + 1$
end

(concrete-)ascending
any n **where**
$n \in P$
$c(n) = c(r)$
then
$c(n) := c(n) + 1$
end

- We have then to prove guard strengthening

Guard Strengthening

$c \in P \to \mathbb{N}$	**inv0_1**
$\forall x, y \cdot c(x) \leq c(y) + 1$	**inv0_2**
$\forall m \cdot \boxed{c(r)} \leq c(m)$	**new invariant**
$n \in P$	Guards of concrete
$\boxed{c(n) = c(r)}$	event ascending
$\vdash$	
$n \in P$	Guards of abstract
$\forall m \cdot \boxed{c(n)} \leq c(m)$	event ascending

Pending Problems

ascending
any n **where**
$n \in P$
$c(n) = c(r)$
then
$c(n) := c(n) + 1$
end

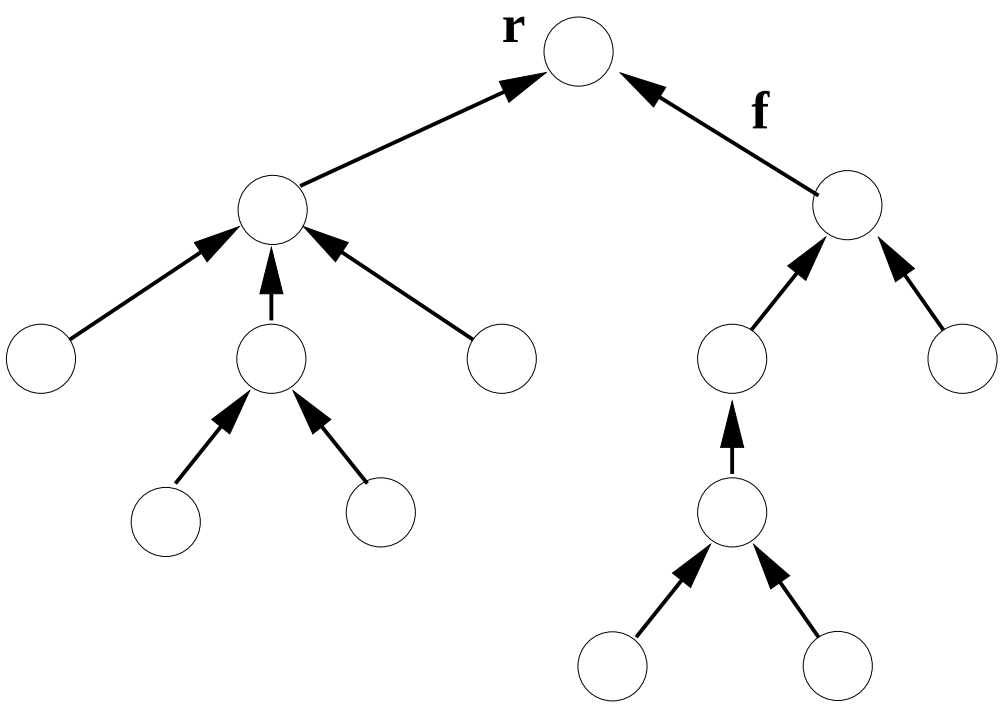
$$\forall m \cdot c(r) \leq c(m)$$

1. We have to prove that the new invariant is preserved by the event
2. The guard of the event still does not fulfill requirement FUN-2

Each node can read the counters of its immediate neighbors only	FUN-2

- Problem 1 solved in this refinement, problem 2 solved later

First Refinement: Let us First Define the Tree



- A tree has got a root r and a parent function f

- This is not sufficient to defined a tree (but enough for the moment)

First Refinement: Defining (Part of) the Tree

- We define the root r of the tree

- And the parent function f (defined everywhere except at the root)

carrier set: P

constants: r, f

axm1_1: $r \in P$

axm1_2: $f \in P \setminus \{r\} \to P$

Achieving Minimal Counter at the Root

- We define a weaker invariant

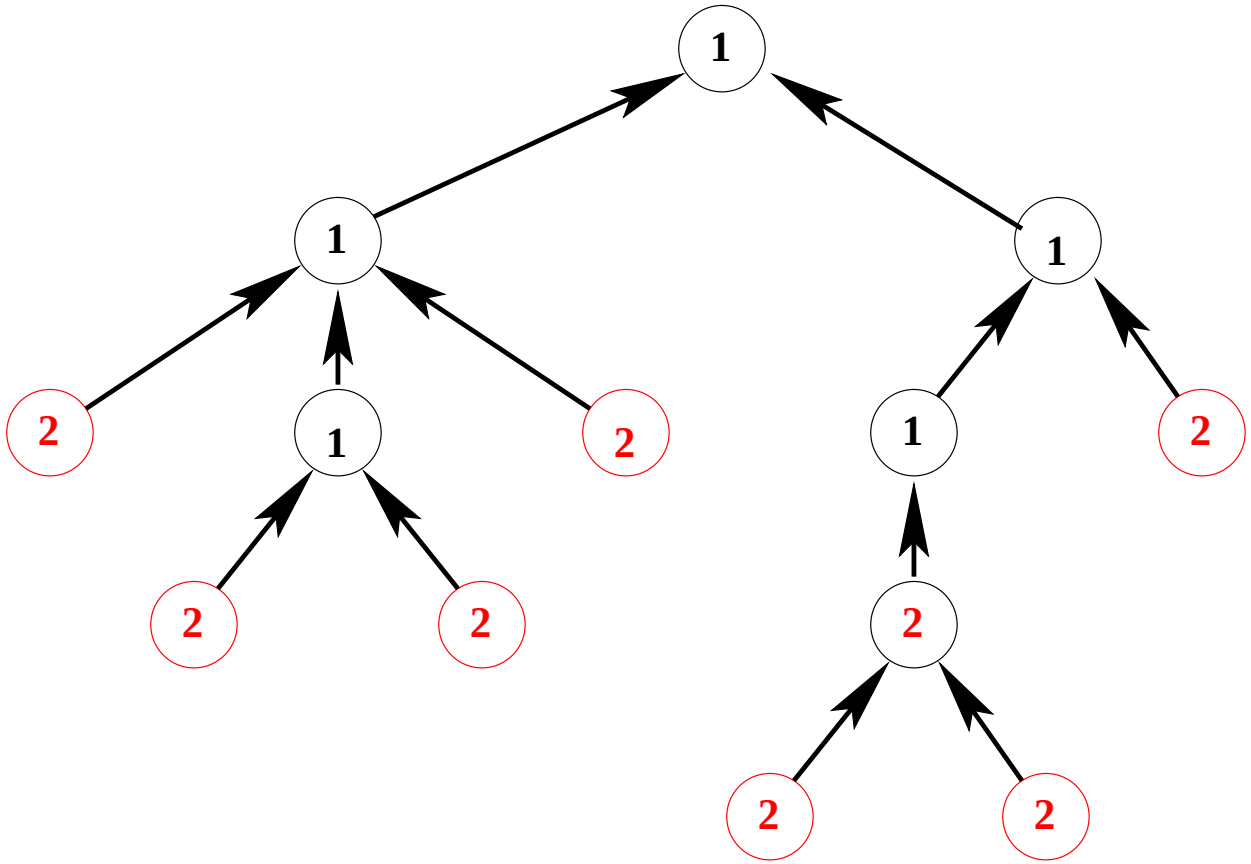
- The counter of the parent of each node m is $\leq$ than that of m

inv1_1: $\forall m \cdot m \neq r \Rightarrow c(f(m)) \leq c(m)$

- The minimality of the counter at the root can be proved:

thm1_1: $\forall m \cdot c(r) \leq c(m)$

Observing the Invariant and Theorem



inv1_1 : $\forall m \cdot m \neq r \Rightarrow c(f(m)) \leq c(m)$

thm1_1 : $\forall m \cdot c(r) \leq c(m)$

A Better Version of Event ascending

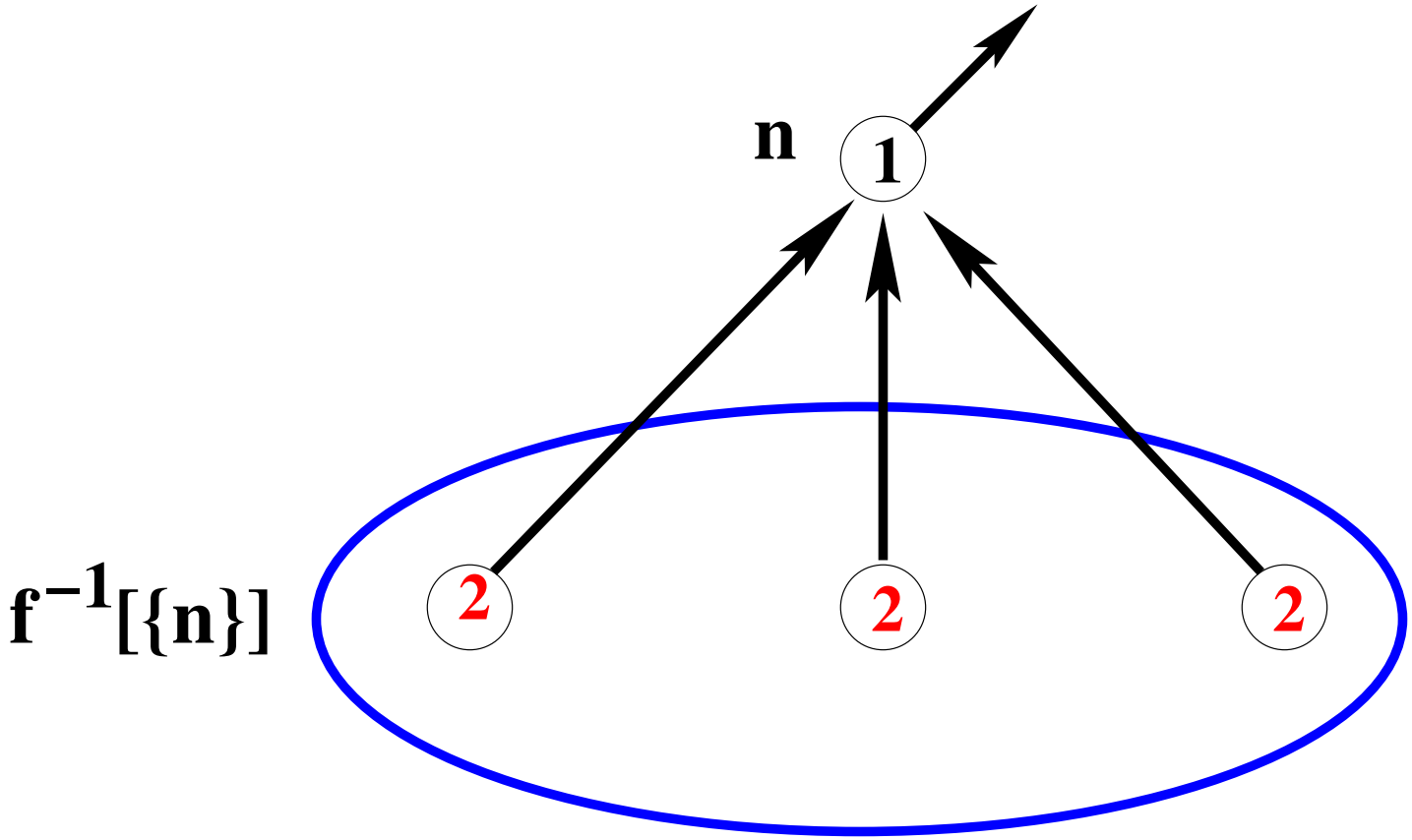
- Adding a guard

ascending
  any n **where**
    $n \in P$
    $c(r) = c(n)$
    $\forall m \cdot m \in f^{-1}[\{n\}] \Rightarrow c(n) < c(m)$
  then
    $c(n) := c(n) + 1$
  end

- This will allow us to prove **inv1_1** easily (again, a proof by cases)

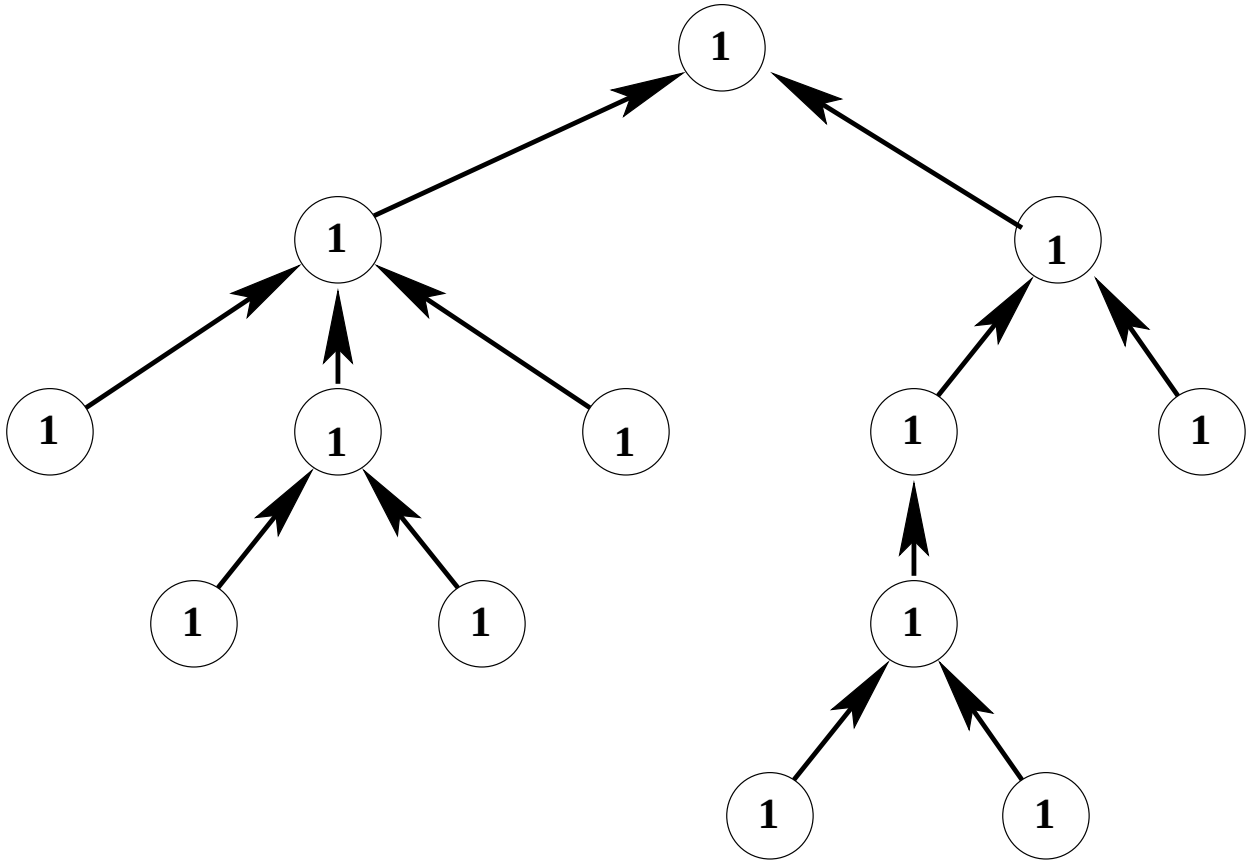
inv1_1 : $\forall m \cdot m \neq r \Rightarrow c(f(m)) \leq c(m)$

The Set of Children of Node n

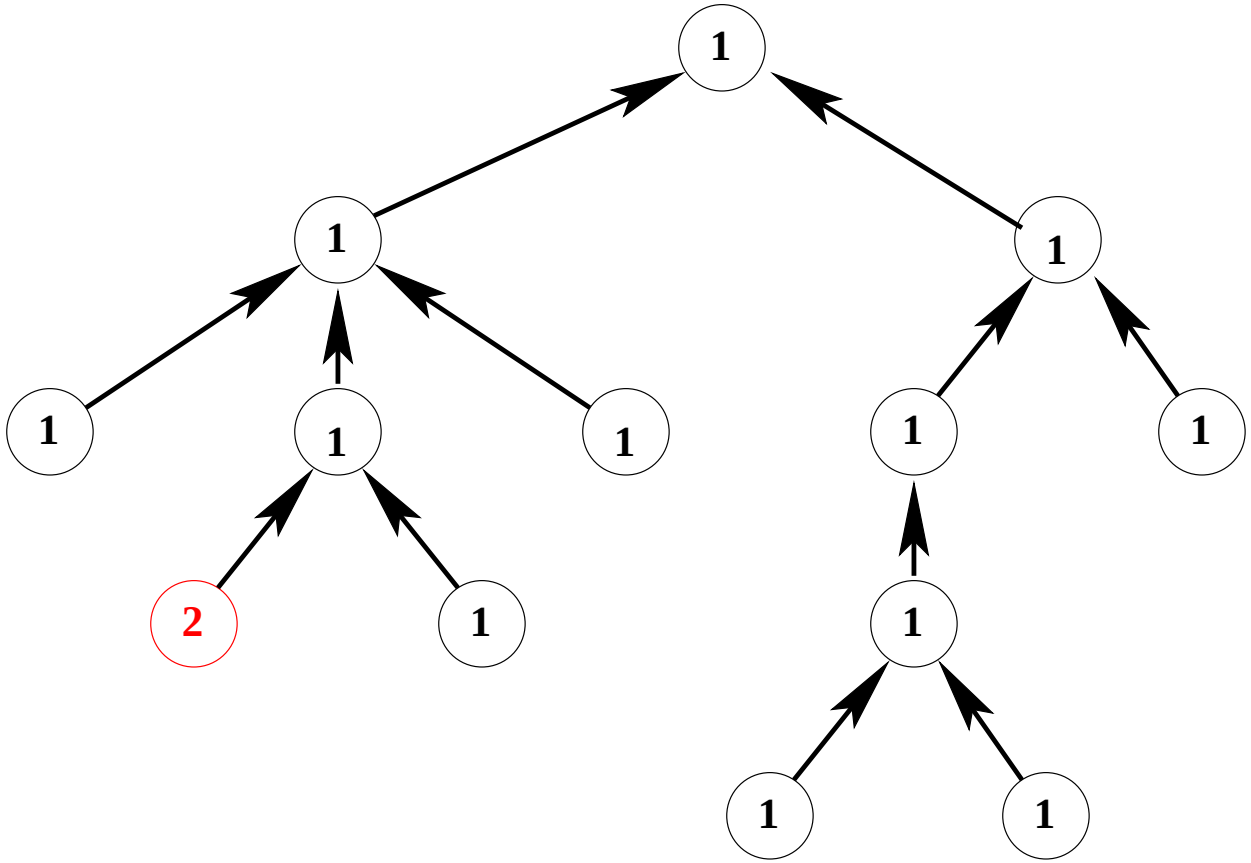


$$\forall m \cdot m \in f^{-1}[\{n\}] \Rightarrow c(n) < c(m)$$

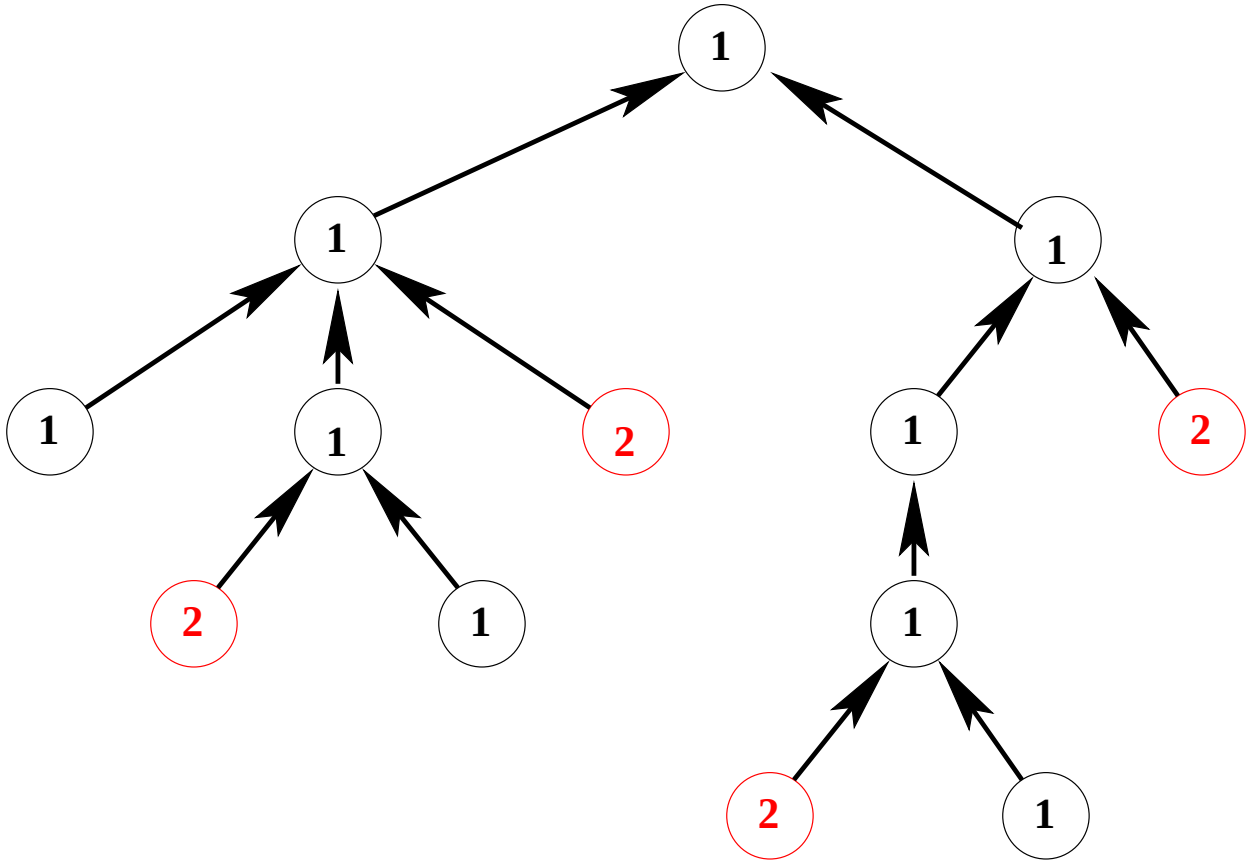
Initial situation



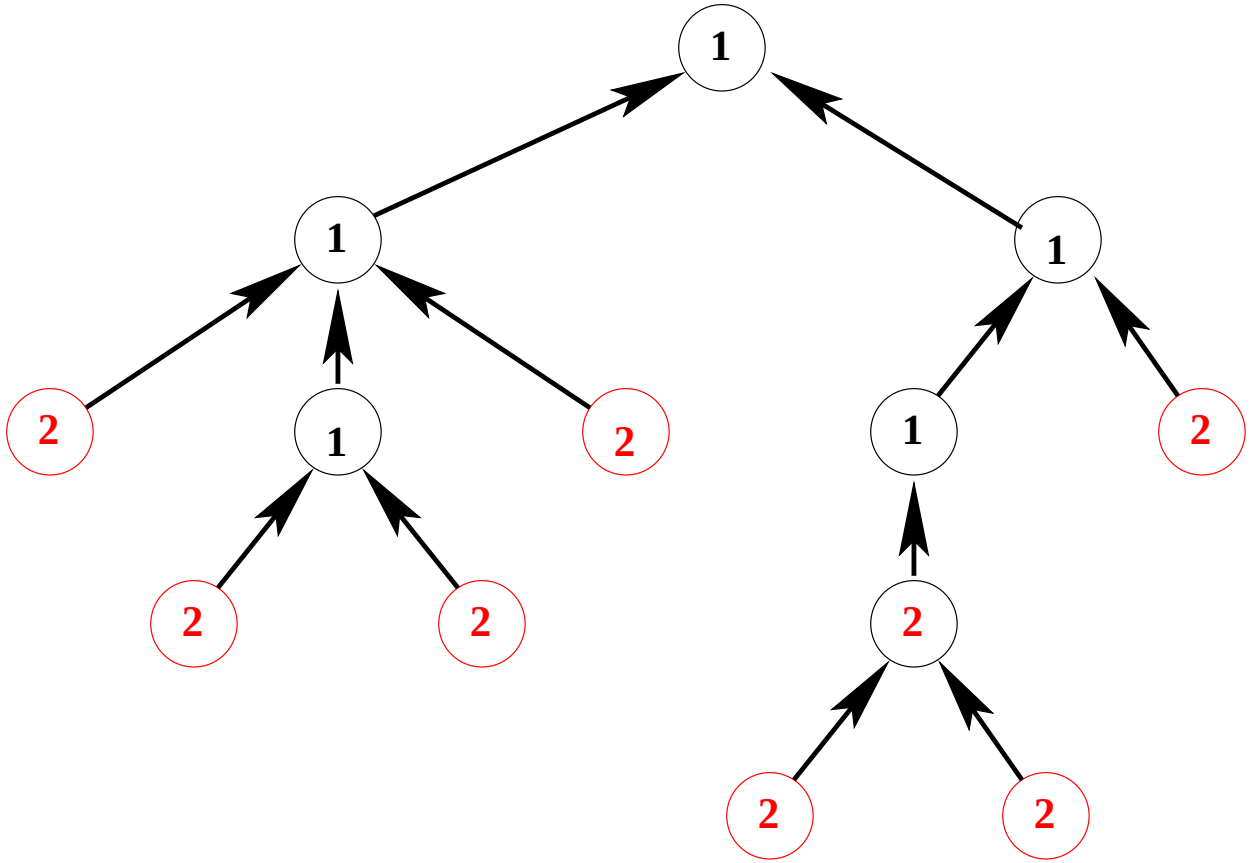
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



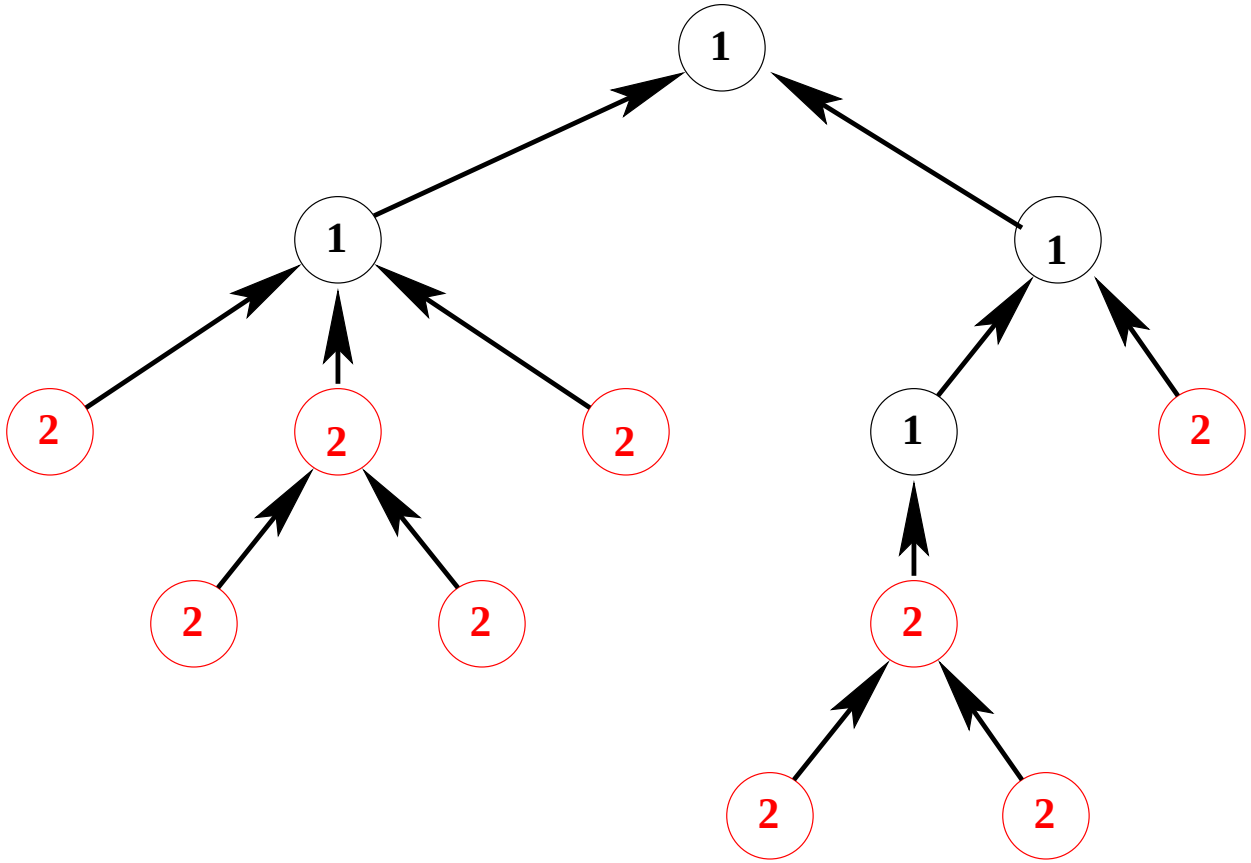
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



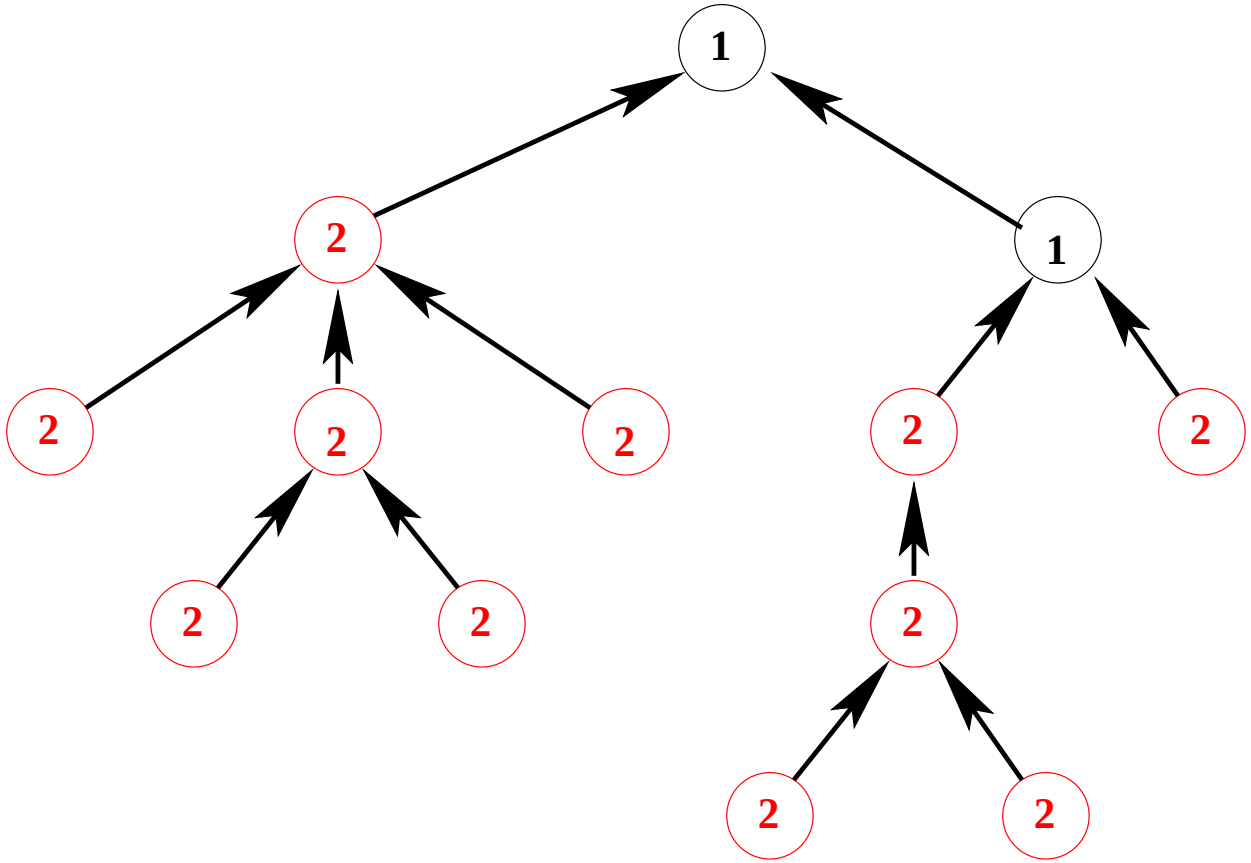
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



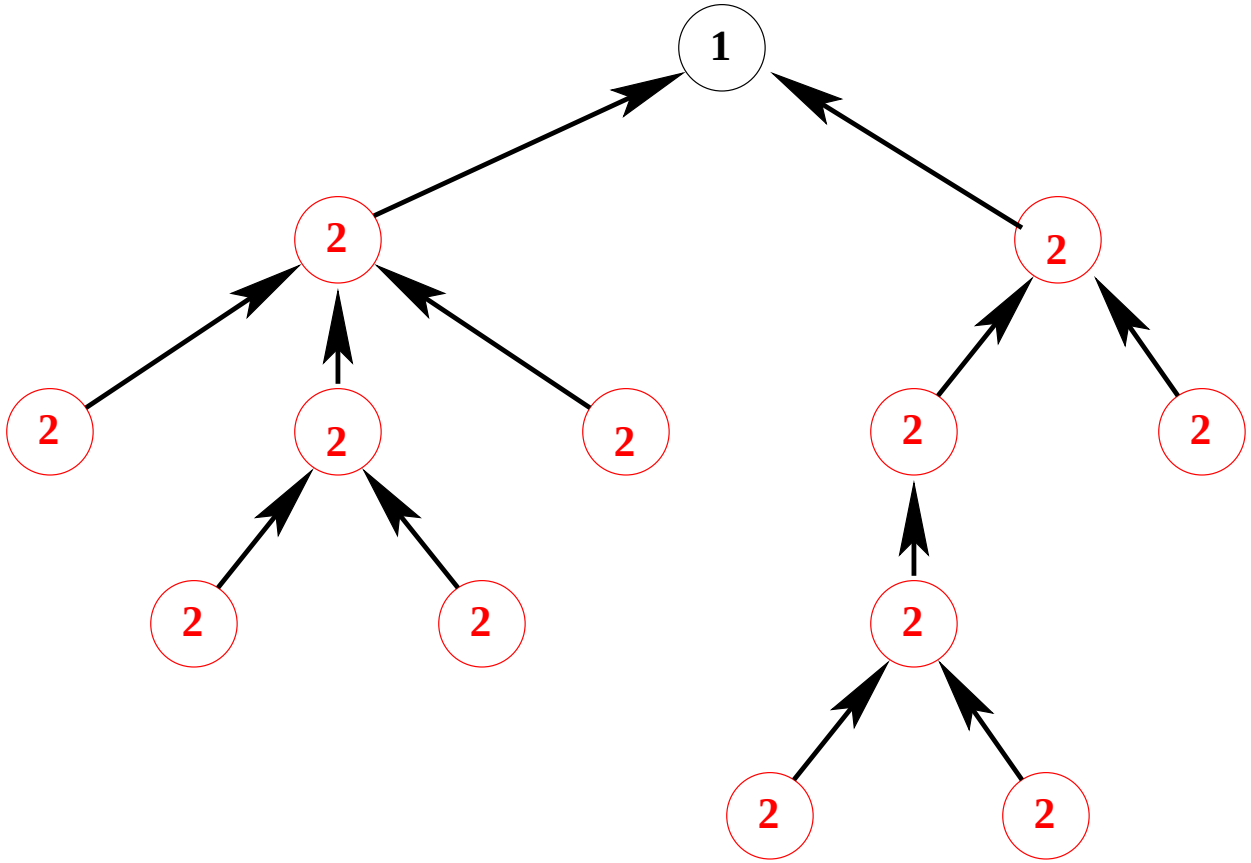
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



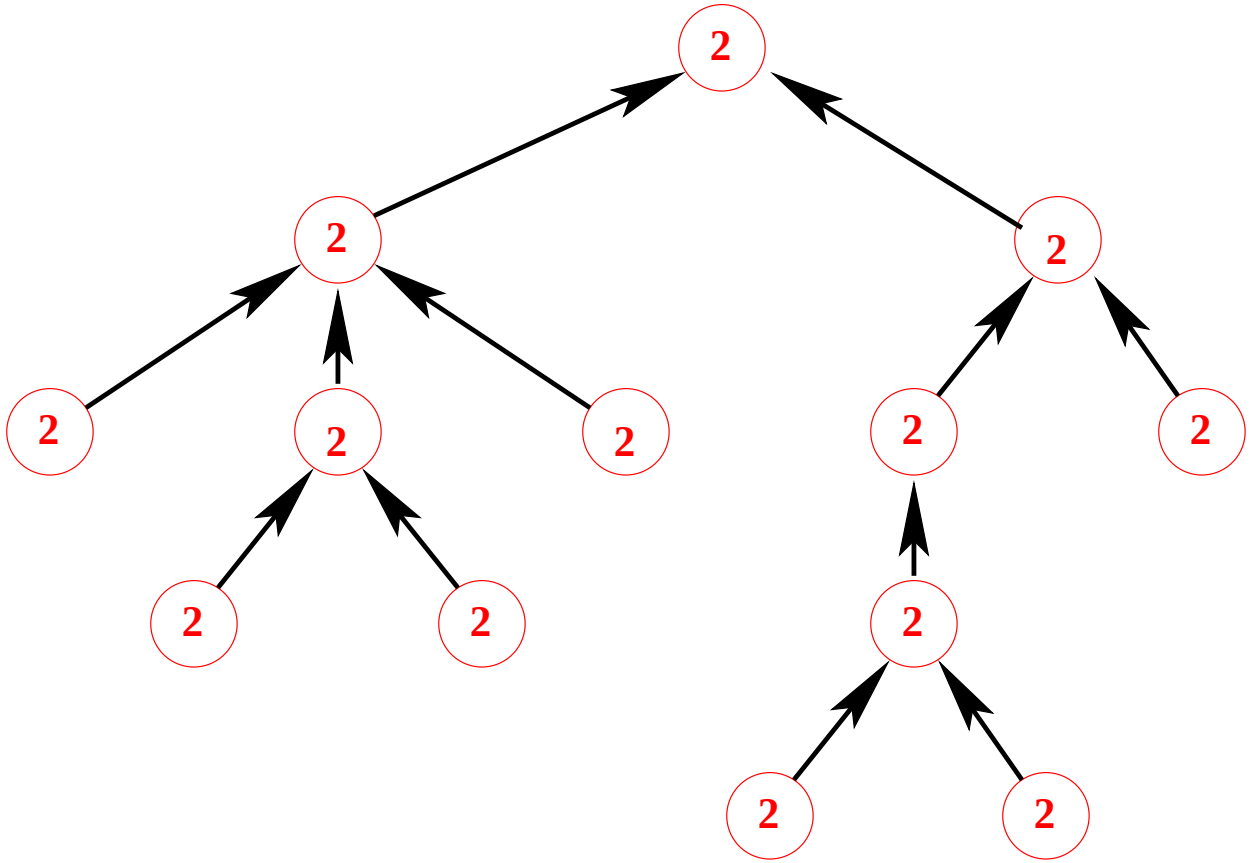
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



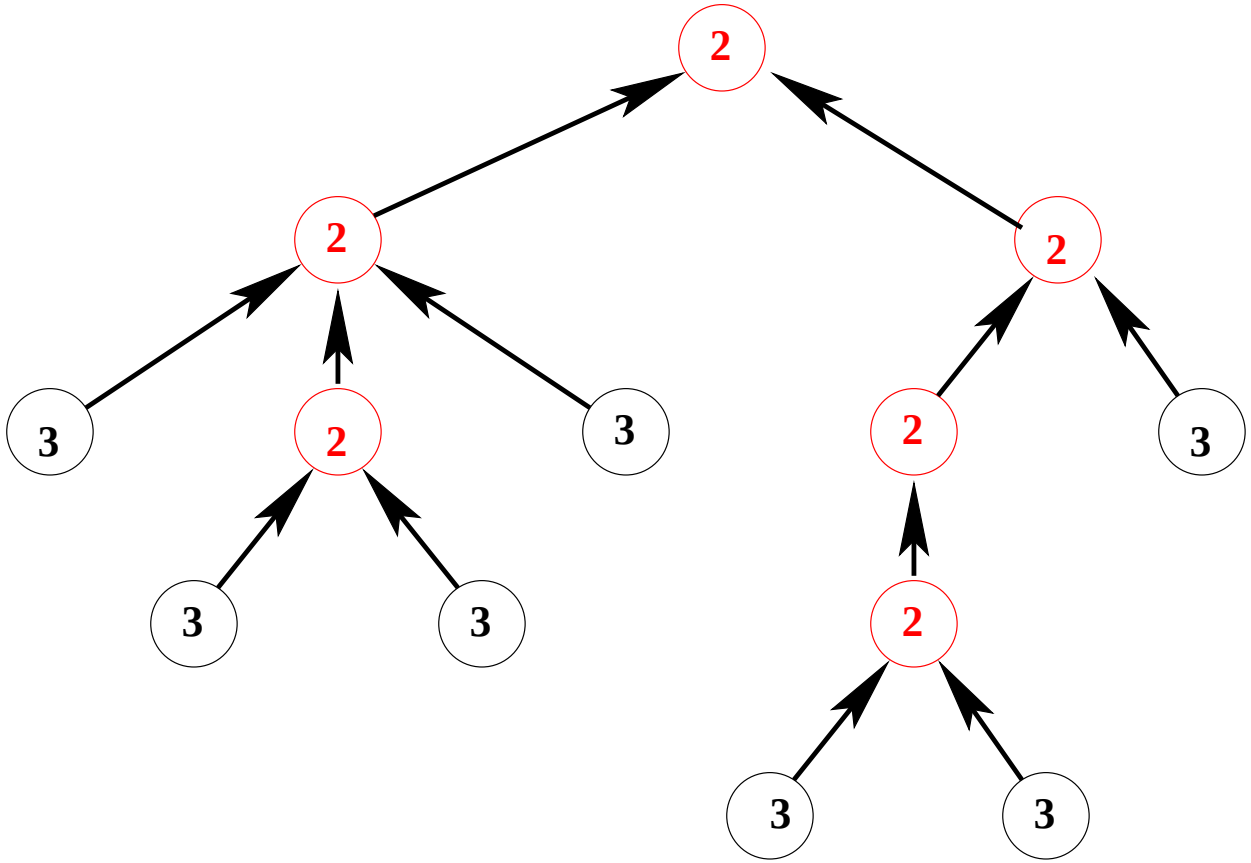
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



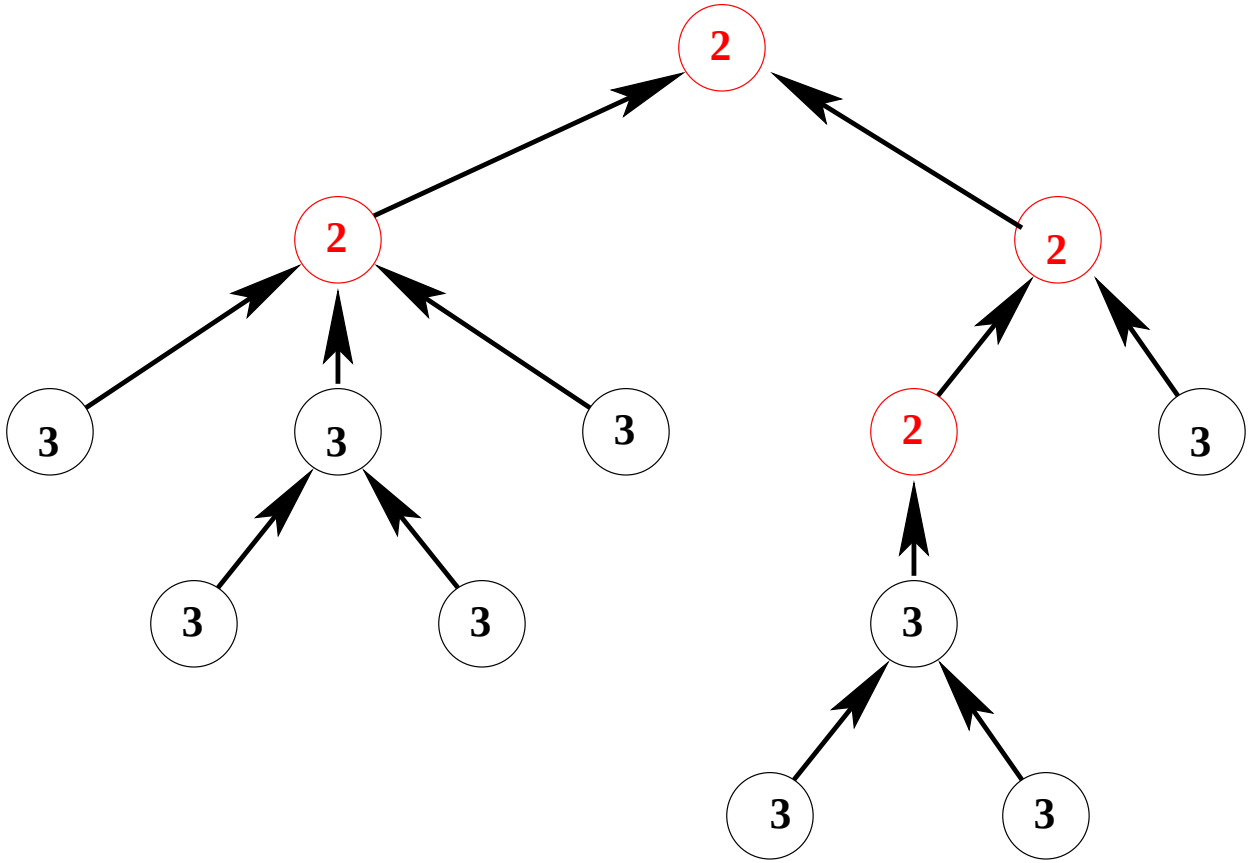
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



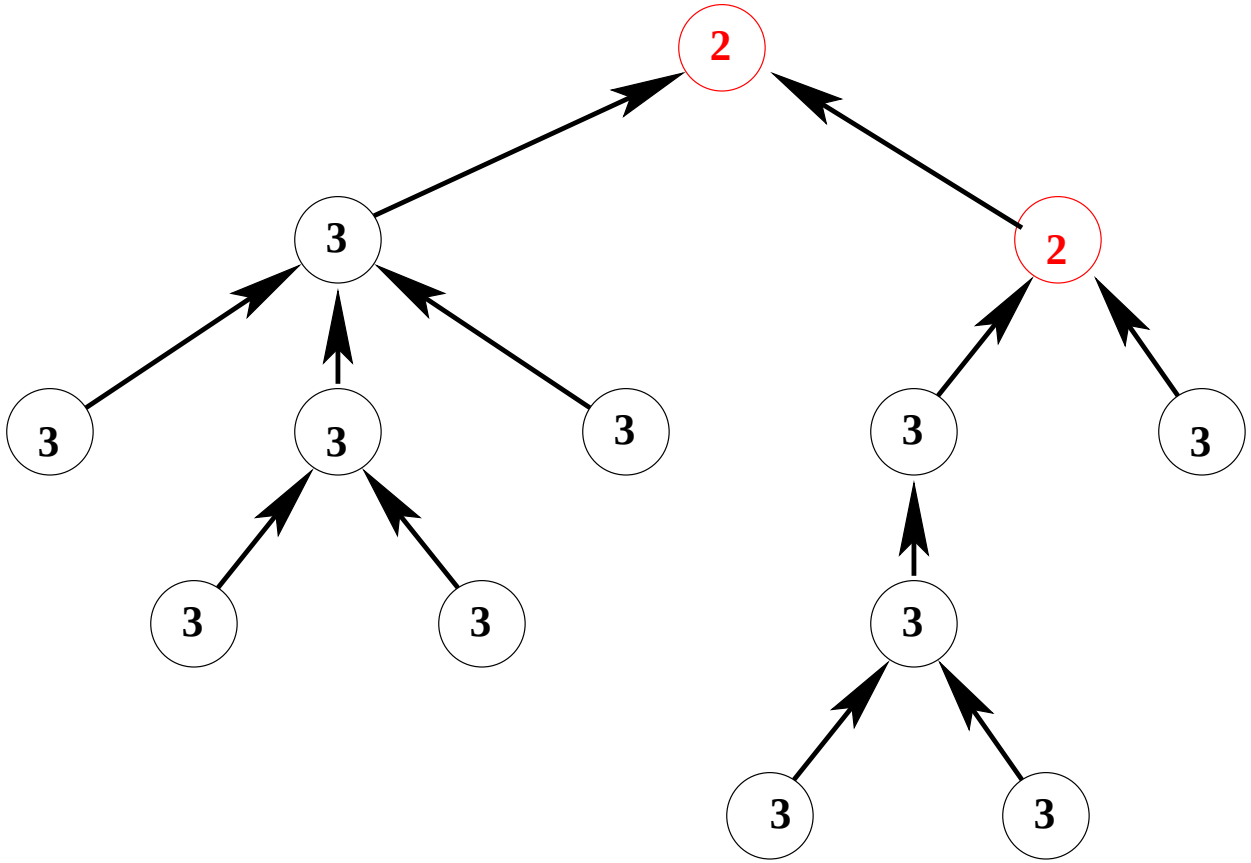
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (m \in f^{-1}[\{n\}] \Rightarrow c(n) < c(m)) \end{cases}$



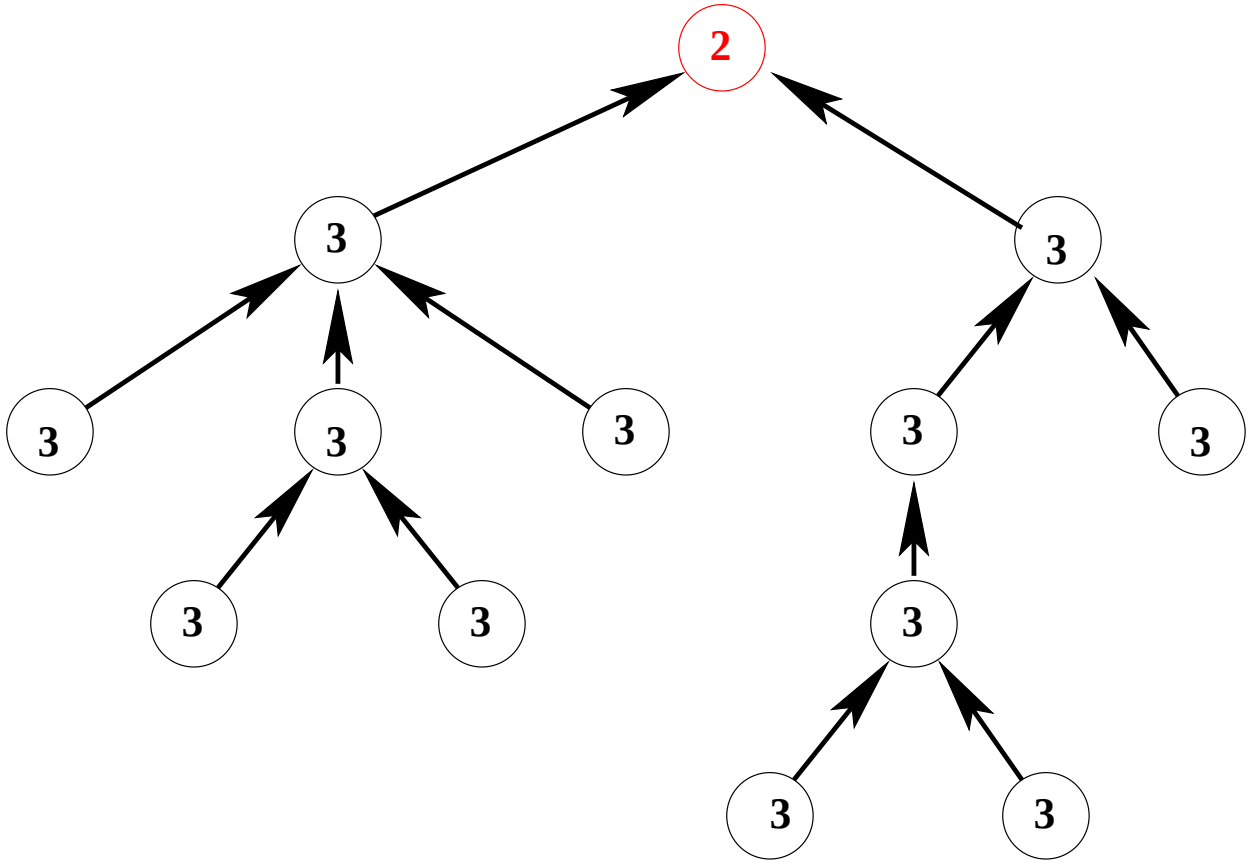
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



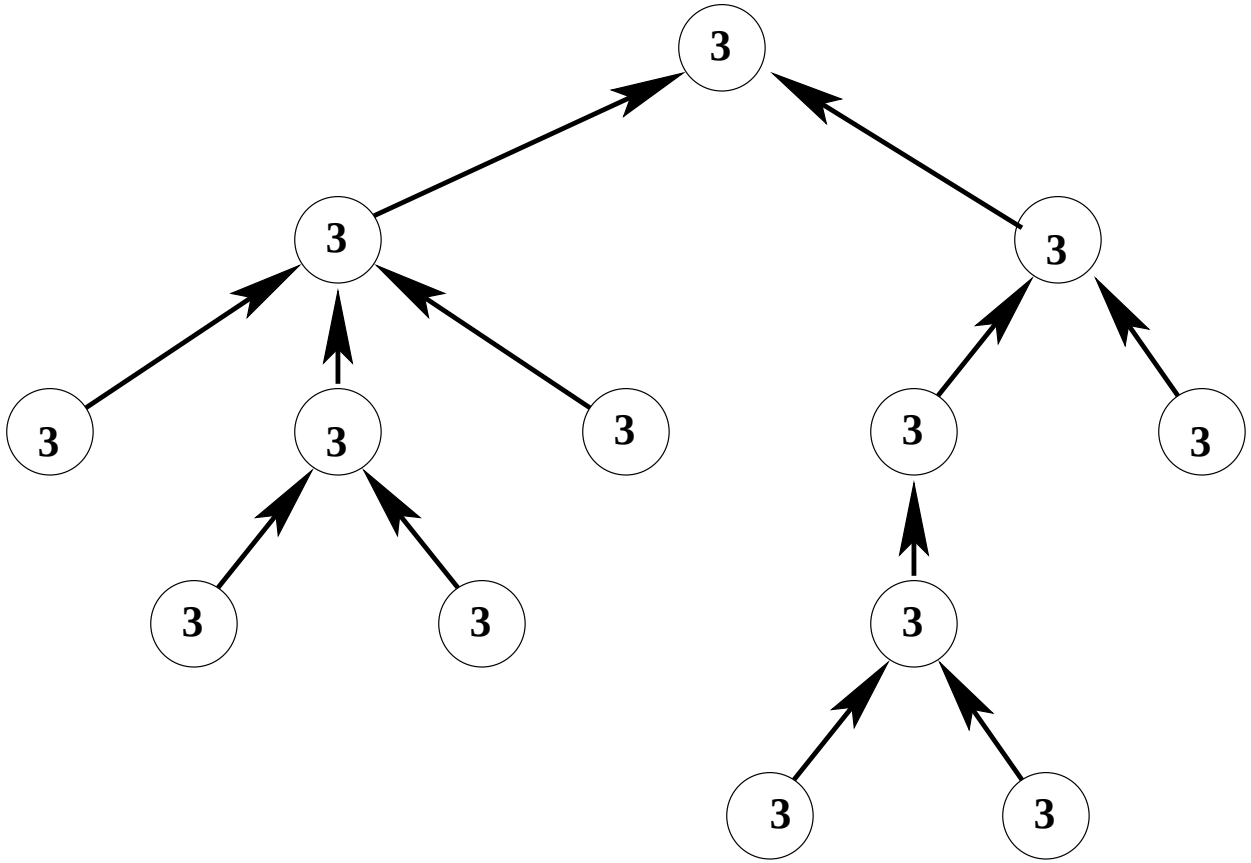
- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$



- the guards: $\begin{cases} c(r) = c(n) \\ \\ \forall m \cdot (\, m \in f^{-1}[\{n\}] \;\Rightarrow\; c(n) < c(m) \,) \end{cases}$

Before Going to Next Refinement, we Must Prove thm1_1

inv0_1: $c \in P \to \mathbb{N}$

inv0_2: $\forall x, y \cdot c(x) \leq c(y) + 1$

axm1_1: $r \in P$

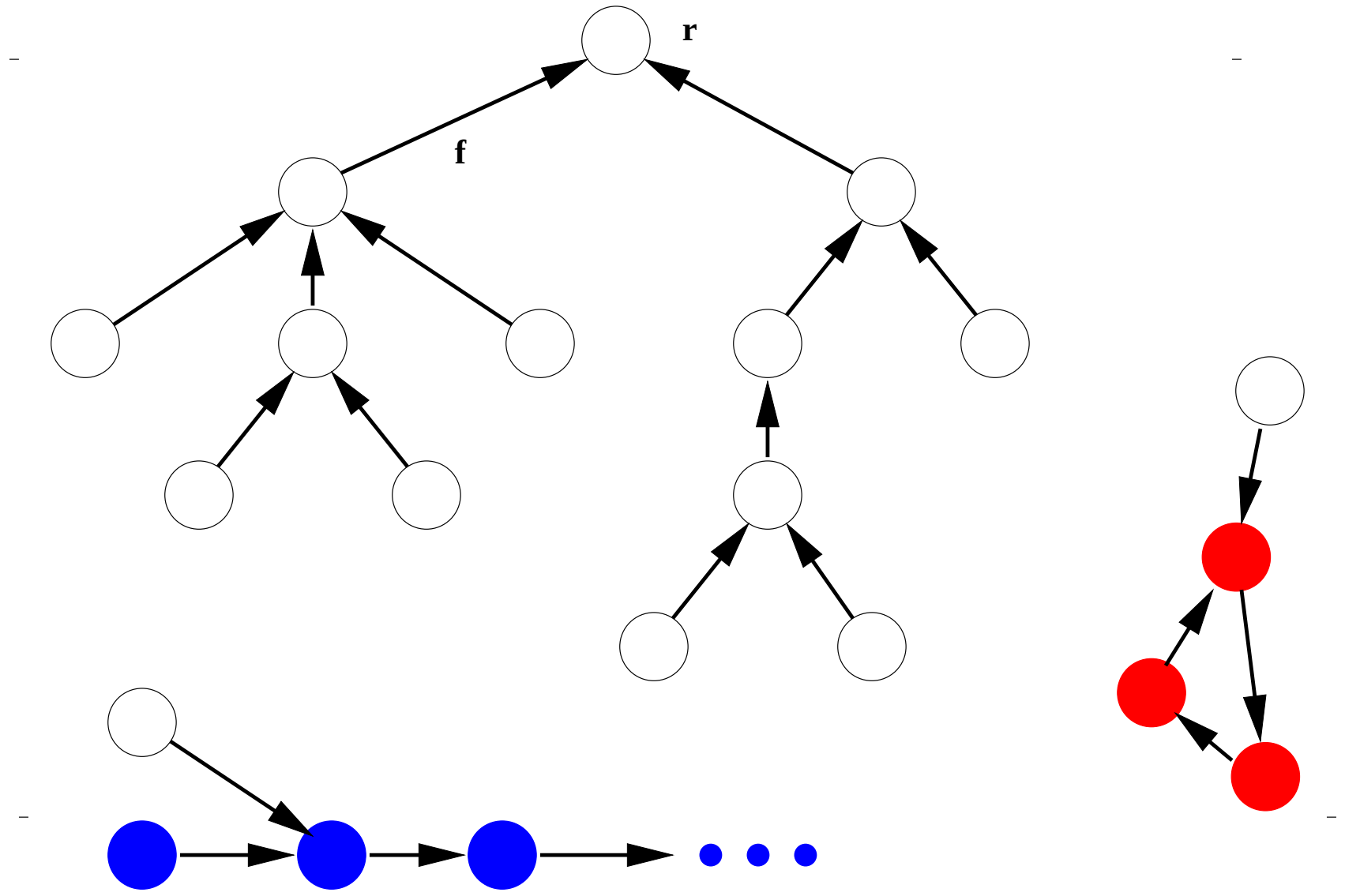
axm1_2: $f \in P \setminus \{r\} \to P$

inv1_1: $\forall m \cdot m \neq r \Rightarrow c(f(m)) \leq c(m)$

thm1_1: $\forall m \cdot c(r) \leq c(m)$

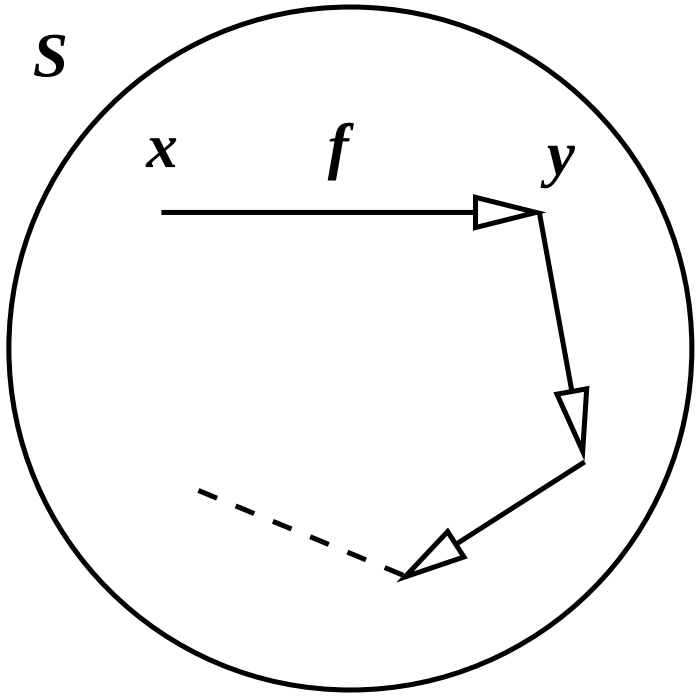
- Properties and invariants are not sufficient to prove **thm1_1**

Problems with the parent Function: Cycles and Infinite Chains



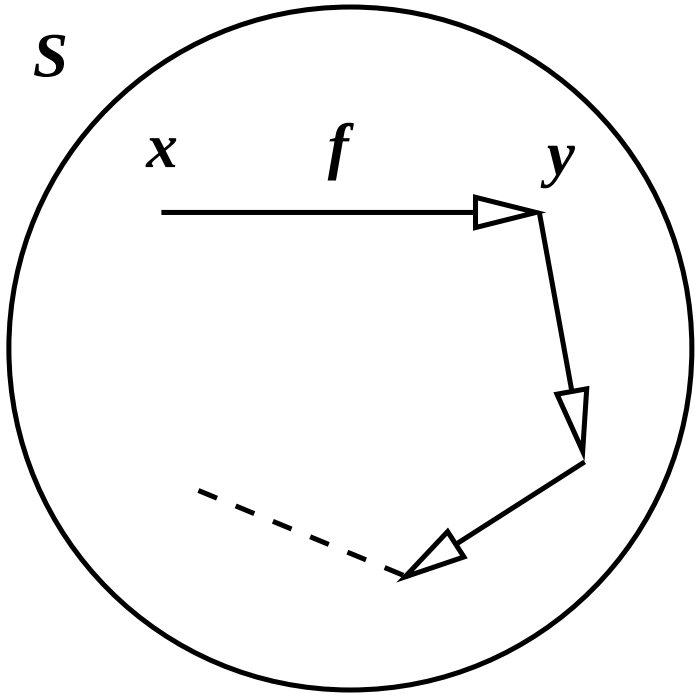
Characterizing Cycles or Infinite Chains in a set S

- The set S is made of cycles or infinite chains



Characterizing Cycles or Infinite Chains in a set S

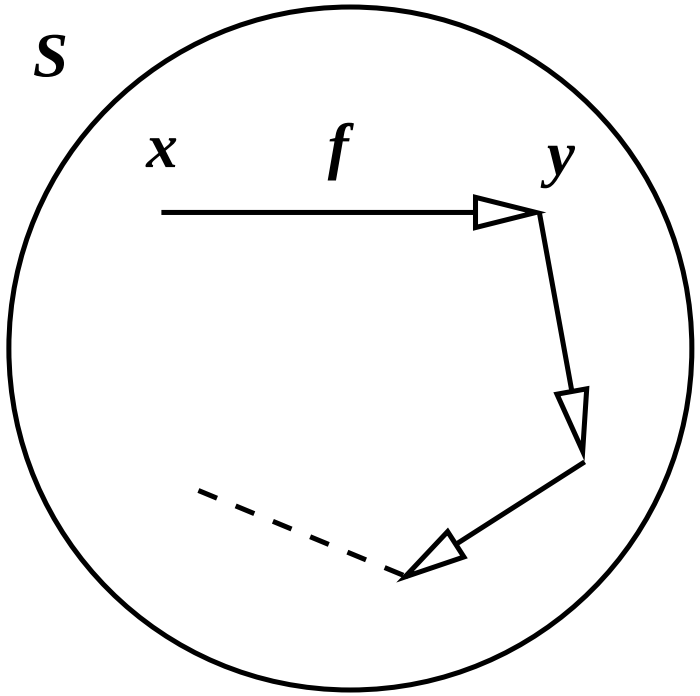
- The set S is made of cycles or infinite chains



$$\forall x \cdot (x \in S \Rightarrow \exists y \cdot (y \in S \land x \mapsto y \in f))$$

Characterizing Cycles or Infinite Chains in a set S

- The set S is made of cycles or infinite chains



$$\forall x \cdot (x \in S \Rightarrow \exists y \cdot (y \in S \wedge x \mapsto y \in f))$$

$$S \subseteq f^{-1}[S]$$

Characterizing Trees

- The root (**axm1_1**)

- The parent function (**axm1_2**)

- There are no cycles and no infinite chains (**axm1_3**)

$$\textbf{axm1\_1}: \quad r \in P$$

$$\textbf{axm1\_2}: \quad f \in P \setminus \{r\} \to P$$

$$\textbf{axm1\_3}: \quad \forall S \cdot S \subseteq f^{-1}[S] \Rightarrow S = \varnothing$$

A Useful Theorem: Tree Induction

axm1_1 : $r \in P$

axm1_2 : $f \in P \setminus \{r\} \to P$

axm1_3 : $\forall S \cdot S \subseteq f^{-1}[S] \Rightarrow S = \varnothing$

thm1_2 : $\forall T \cdot r \in T \land f^{-1}[T] \subseteq T \Rightarrow P \subseteq T$

DEMO

A Tree

axm1_1 : $r \in P$ Root

axm1_2 : $f \in P \setminus \{r\} \to P$ Parent function

thm1_2 : $\forall T \cdot r \in T \ \land \ f^{-1}[T] \subseteq T \ \Rightarrow \ P \subseteq T$

DEMO

$$\textbf{thm1\_1}: \quad \forall m \cdot c(r) \leq c(m)$$

Back to the Problem with First Refinement

ascending
  any n **where**
    $n \in P$
    $c(r) = c(n)$
    $\forall\, m \cdot m \in f^{-1}[\{n\}] \Rightarrow c(n) < c(m)$
  then
    $c(n) := c(n) + 1$
  end

- The third guard is correct (n uses its children counters only)

- The second guard is not correct (n uses the root counter)

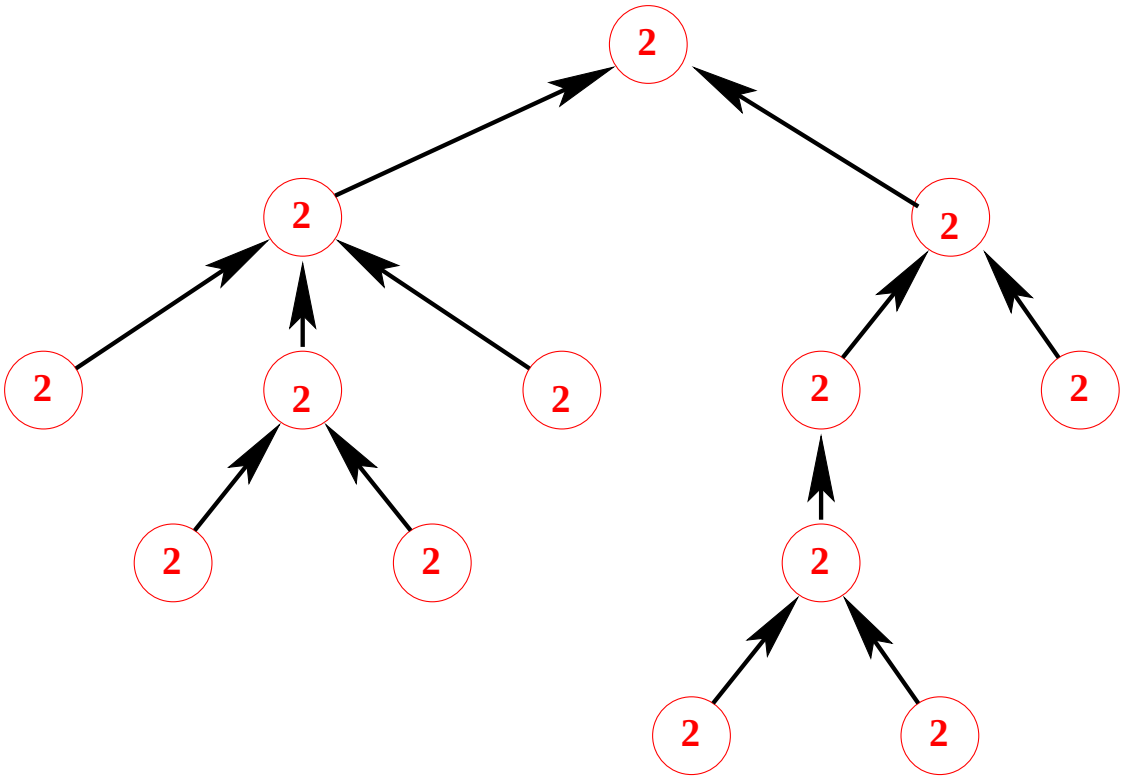
Still a Problem

ascending
  any n **where**
    $n \in P$
    $c(r) = c(n)$
    $\forall m \cdot m \in f^{-1}[\{n\}] \Rightarrow c(n) < c(m)$
  then
    $c(n) := c(n) + 1$
  end

- The second guard is not correct: n uses the root counter $c(r)$

Second Refinement

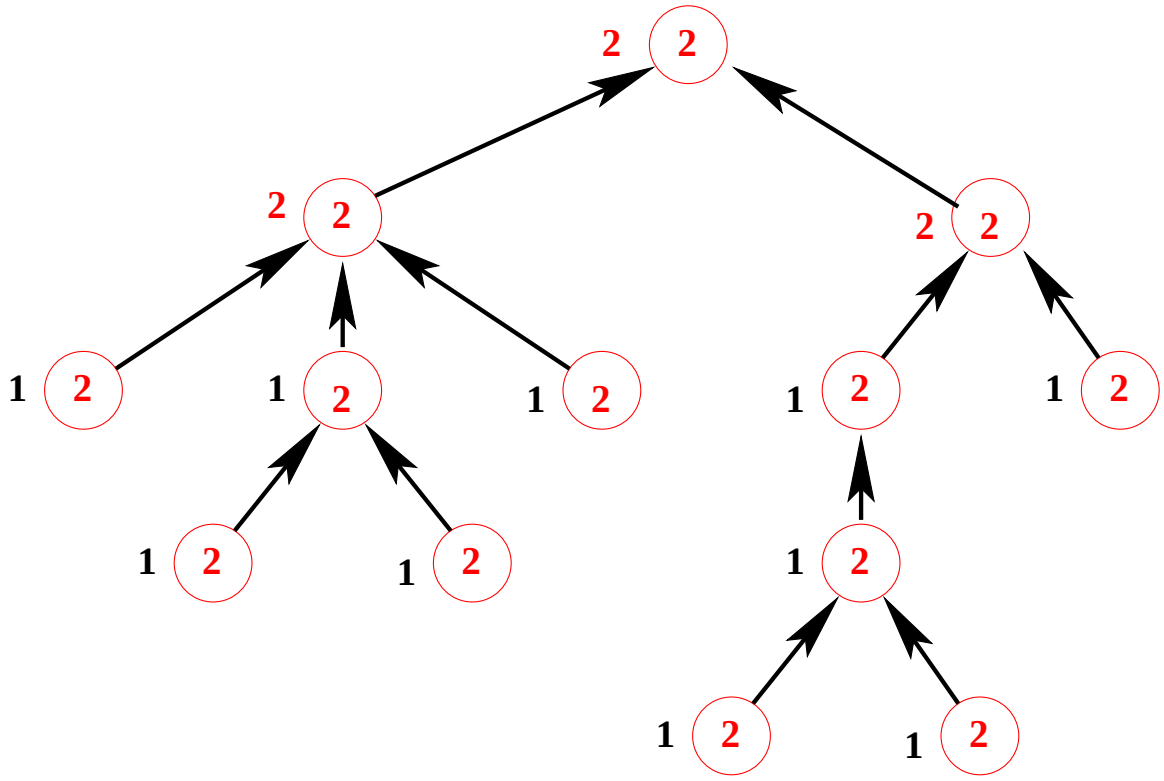
- We want to replace the guard $c(r) = c(n)$ in event "ascending"



- Processes must be aware when this situation does occur

Introducing a Second Counter at each Node

We add a second counter d at each node



- The second counter d has properties which are similar to those of c

Second Refinement: the State

carrier set: P

constants: r, f

variables: c, d

Invariant **inv2_2** is as **inv0_2**

inv2_1: $d \in P \to \mathbb{N}$

inv2_2: $\forall x, y \cdot d(x) \leq d(y) + 1$

Second Refinement: the Events

ascending
any n **where**
$n \in P$
$c(r) = c(n)$
$\forall m \cdot m \in f^{-1}[\{n\}] \Rightarrow c(n) < c(m)$
then
$c(n) := c(n) + 1$
end

descending
any n **where**
$n \in P$
$\forall m \cdot d(n) \leq d(m)$
then
$d(n) := d(n) + 1$
end

- Proof of the preservation of **inv2_2** by event "descending" is easy

inv2_2: $\forall x, y \cdot d(x) \leq d(y) + 1$

- It is similar to that of the preservation of **inv0_2** by event "ascending"

Third Refinement: Tasks

- We extend the invariant of counter d

- We establish the relationship between both counters c and d

- This will allow us to refine event ascending

- We construct the descending wave (by refining event descending)

- Remark: this is the most difficult refinement

Third Refinement: Part of the State

inv3_1: $\forall m \cdot m \neq r \Rightarrow d(m) \leq d(f(m))$

inv3_2: $d(r) \leq c(r)$

thm3_1: $\forall m \cdot d(m) \leq d(r)$

- **thm3_1** can be proved by using the tree Induction (**axm1_3**)

- **inv3_1** and **thm3_1** have to be compared to **inv1_1** and **thm1_1**

inv1_1: $\forall m \cdot m \neq r \Rightarrow c(f(m)) \leq c(m)$

thm1_1: $\forall m \cdot c(r) \leq c(m)$

Refining Event ascending: Guard Strengthening

(abstract-)ascending
any n **where**
$n \in P$
$c(n) = c(r)$
...
then
$c(n) := c(n) + 1$
end

(concrete-)ascending
any n **where**
$n \in P$
$c(n) = d(n)$
...
then
$c(n) := c(n) + 1$
end

concrete guard
according to **thm3_1**
invariant **inv3_2**
according to **thm1_1**

abstract guard

$c(n) = d(n)$
$d(n) \leq d(r)$
$d(r) \leq c(r)$
$c(r) \leq c(n)$
$\vdash$
$c(n) = c(r)$

- We have reached our goal: event ascending indeed fulfills FUN-2

Refining Event descending (First Case)

(abstract-)descending
any n **where**
$n \in P$
$\forall m \cdot d(n) \leq d(m)$
then
$d(n) := d(n) + 1$
end

(concrete-)descending_1
any n **where**
$n \in P \setminus \{r\}$
$d(n) \neq d(f(n))$
then
$d(n) := d(n) + 1$
end

Guard strengthening:

$$
\begin{array}{l}
n \in P \setminus \{r\} \\
d(n) \neq d(f(n)) \\
\Rightarrow \\
d(n) \leq d(m)
\end{array}
$$

Additional Theorems Needed

- In order to prove guard strengthening, we need the theorems:

thm3_2: $\forall n \cdot n \neq r \;\Rightarrow\; d(f(n)) \;\in\; d(n) \,..\, d(n)+1\,)$

thm3_3: $\forall n \cdot d(r) \;\in\; d(n) \,..\, d(n)+1$

Refining Event descending (Second Case)

(abstract-)descending
any n **where**
$n \in P$
$\forall m \cdot d(n) \leq d(m)$
then
$d(n) := d(n) + 1$
end

(concrete-)descending_2
when
$d(r) \neq c(r)$
then
$d(r) := d(r) + 1$
end

- Here we need a witness for n: the root r is the obvious choice

Guard strengthening

$$d(r) \neq c(r)$$
$$m \in P$$
$$\Rightarrow$$
$$d(r) \leq d(m)$$

Additional Theorem Needed

- In order to prove guard strengthening,we need the theorem

thm3_4: $\forall n \cdot c(r) \in d(n) \mathinner{..} d(n) + 1$

Additional Invariant Needed

- In order to prove the previous theorem

thm3_4: $\forall n \cdot c(r) \in d(n) \mathinner{..} d(n) + 1$

- We need the following additional invariant

inv3_3: $\forall n \cdot c(n) \in d(n) \mathinner{..} d(n) + 1$

- We have thus to prove that this invariant is preserved by the three events: ascending, descending_1, and descending_2.

Summary of Third Refinement: the State

inv3_1: $\forall m \cdot m \neq r \Rightarrow d(m) \leq d(f(m))$

inv3_2: $d(r) \leq c(r)$

inv3_3: $\forall n \cdot c(n) \in d(n) \mathinner{..} d(n)+1$

thm3_1: $\forall m \cdot d(m) \leq d(r)$

thm3_2: $\forall n \cdot n \neq r \Rightarrow d(f(n)) \in d(n) \mathinner{..} d(n)+1$

thm3_3: $\forall n \cdot d(r) \in d(n) \mathinner{..} d(n)+1$

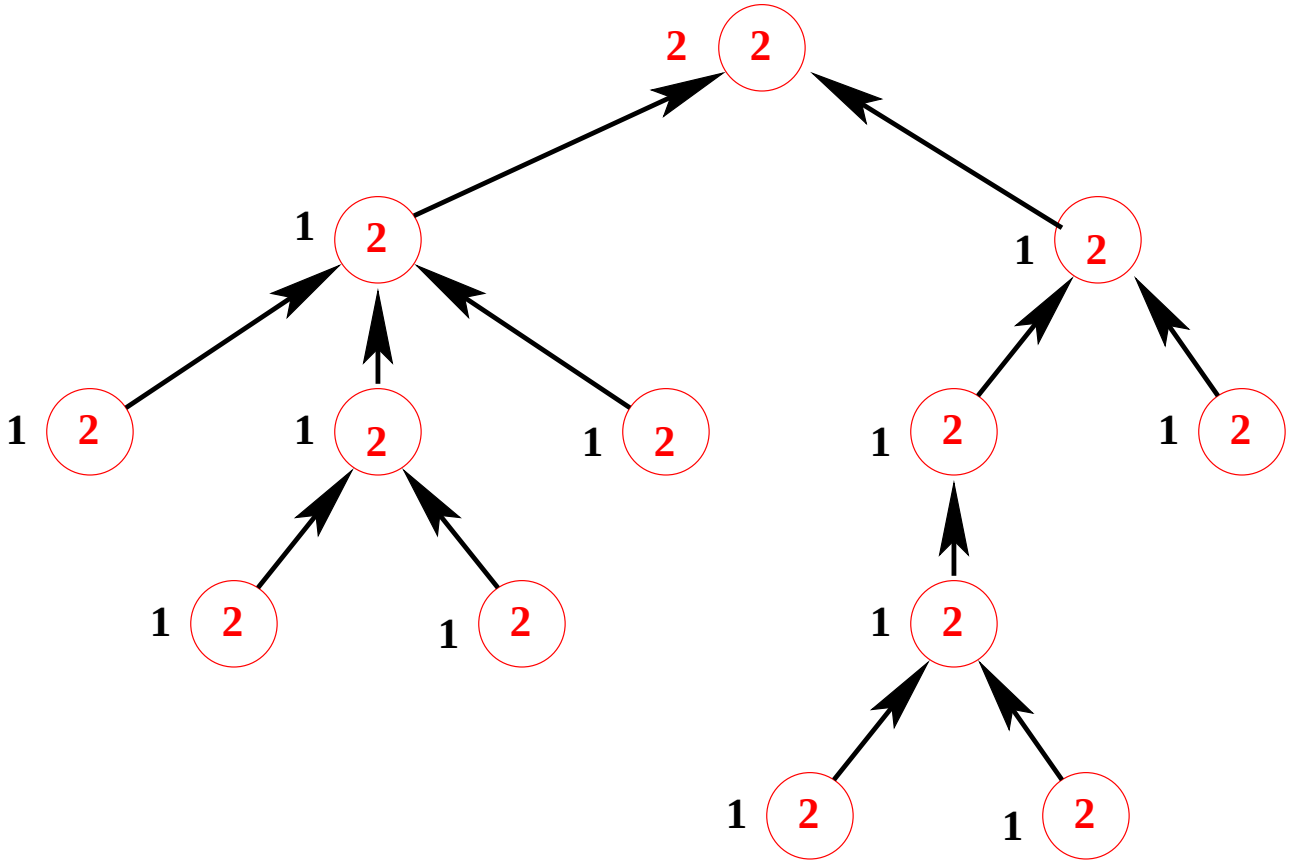
thm3_4: $\forall n \cdot c(r) \in d(n) \mathinner{..} d(n)+1$

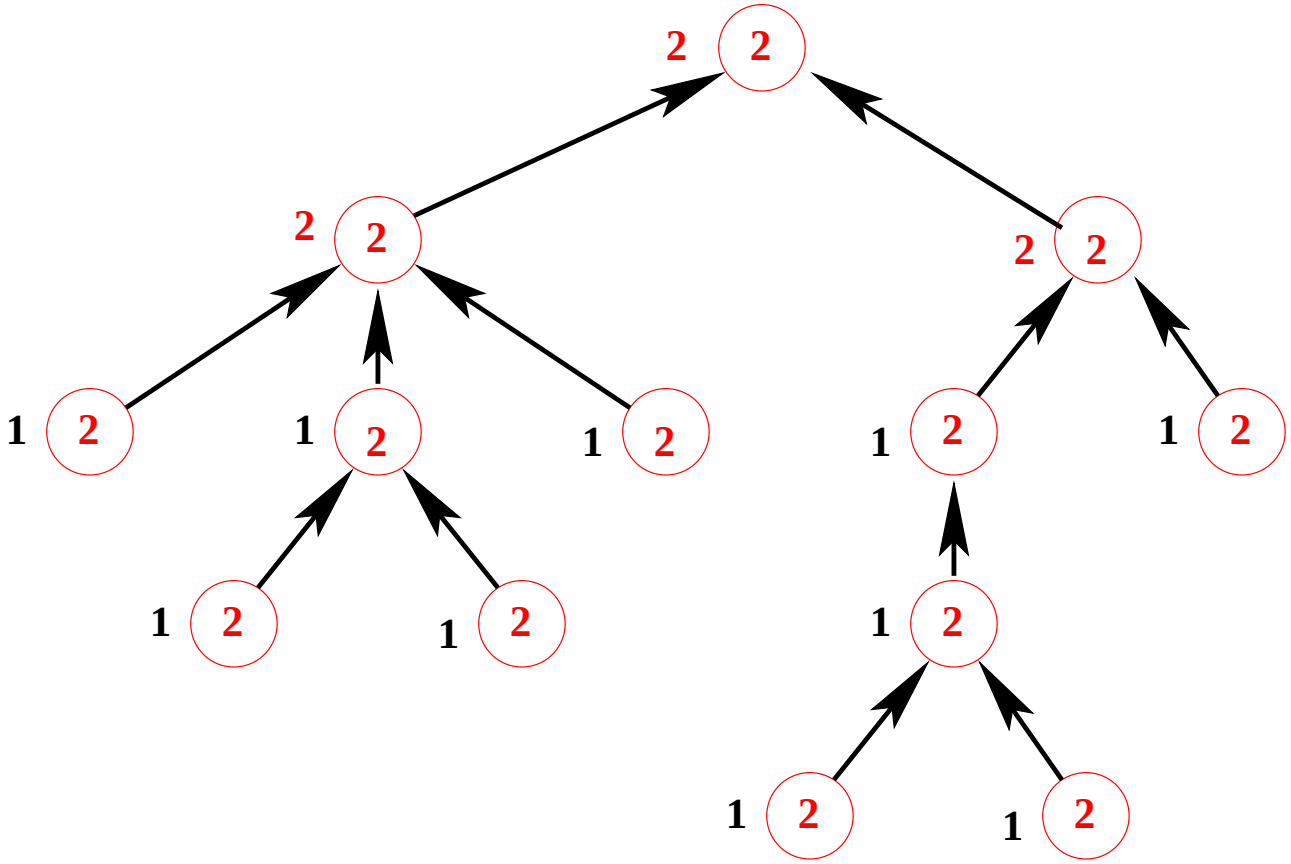
Summary of Third Refinement: the Events

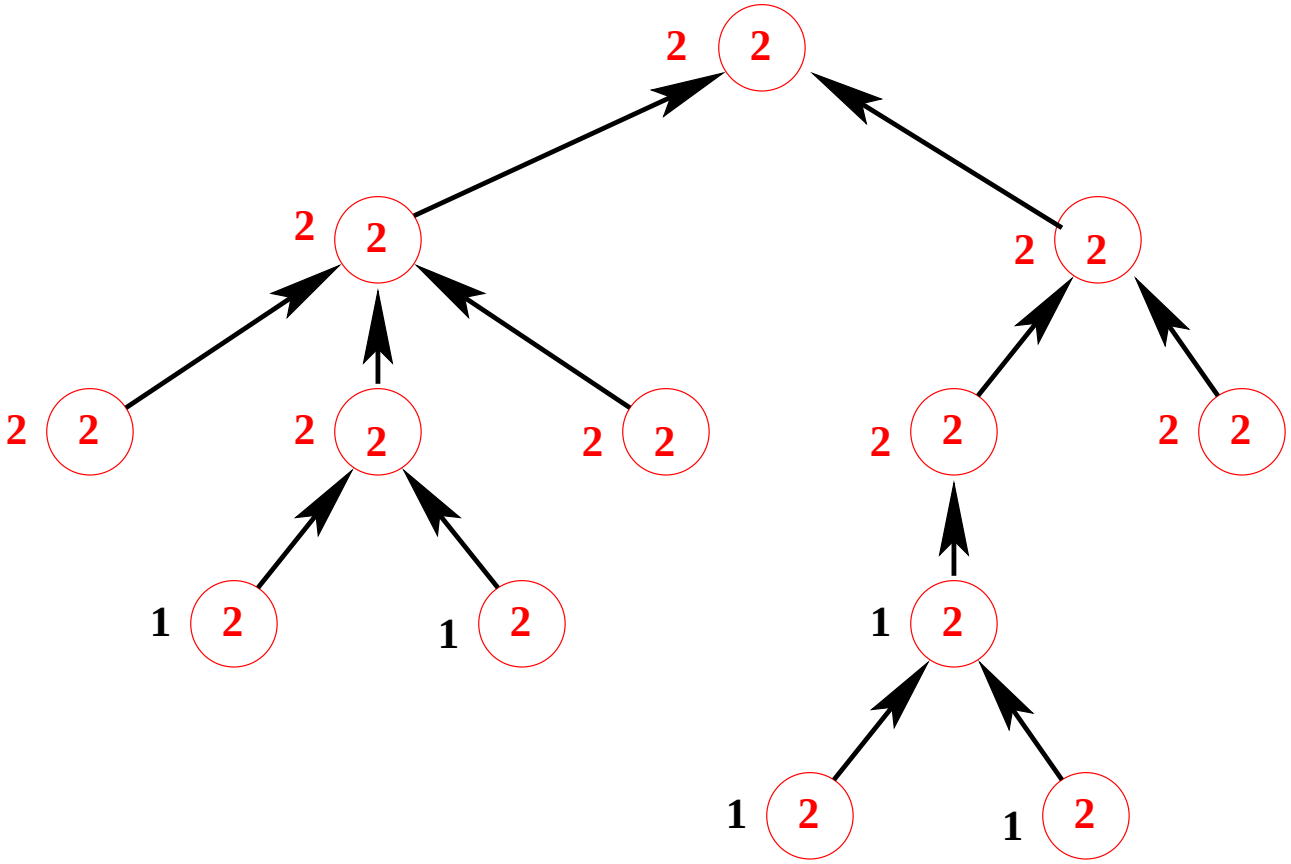
ascending
any n **where**
$n \in P$
$c(n) = d(n)$
$\forall m \cdot (m \in f^{-1}[\{n\}] \Rightarrow c(n) \neq c(m))$
then
$c(n) := c(n) + 1$
end

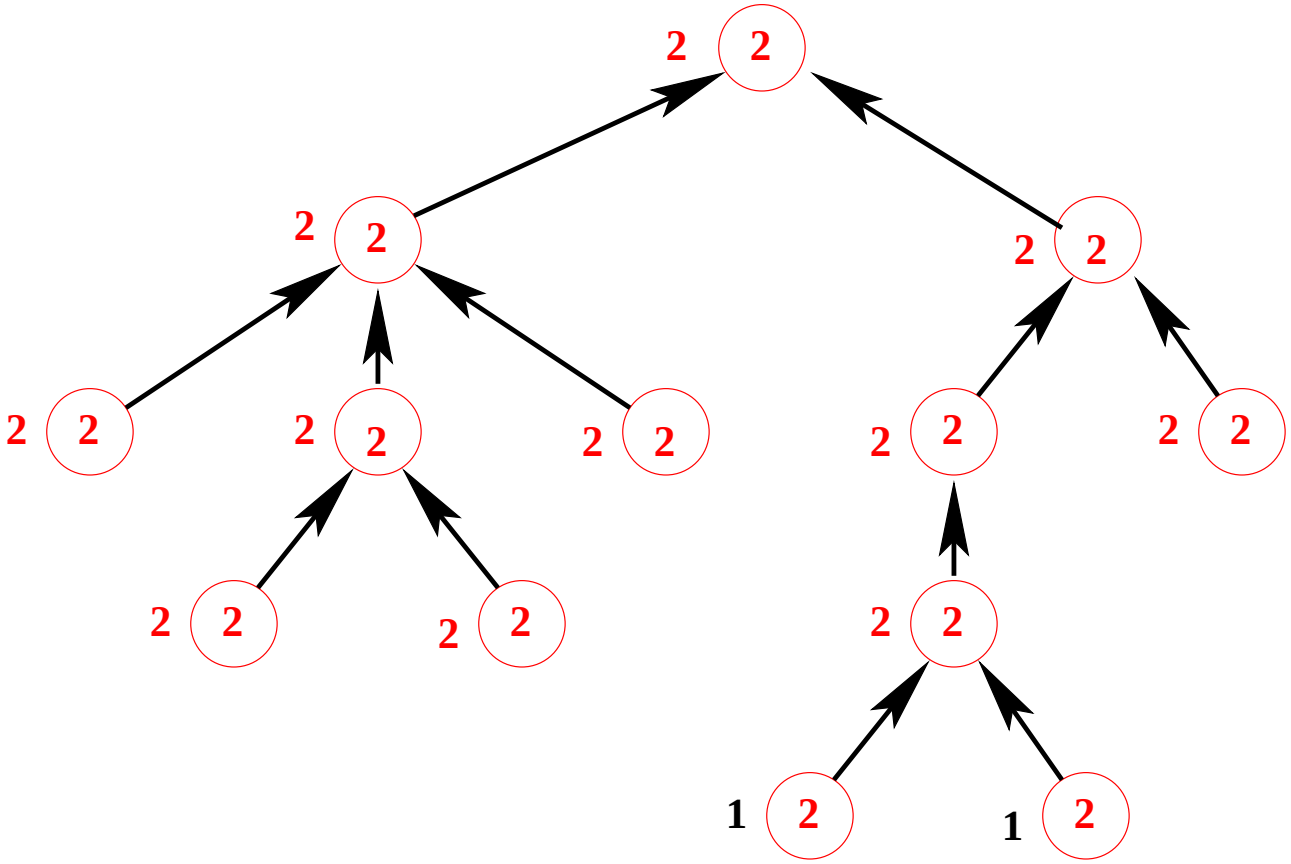
descending_1
any n **where**
$n \in P \setminus \{r\}$
$d(n) \neq d(f(n))$
then
$d(n) := d(n) + 1$
end

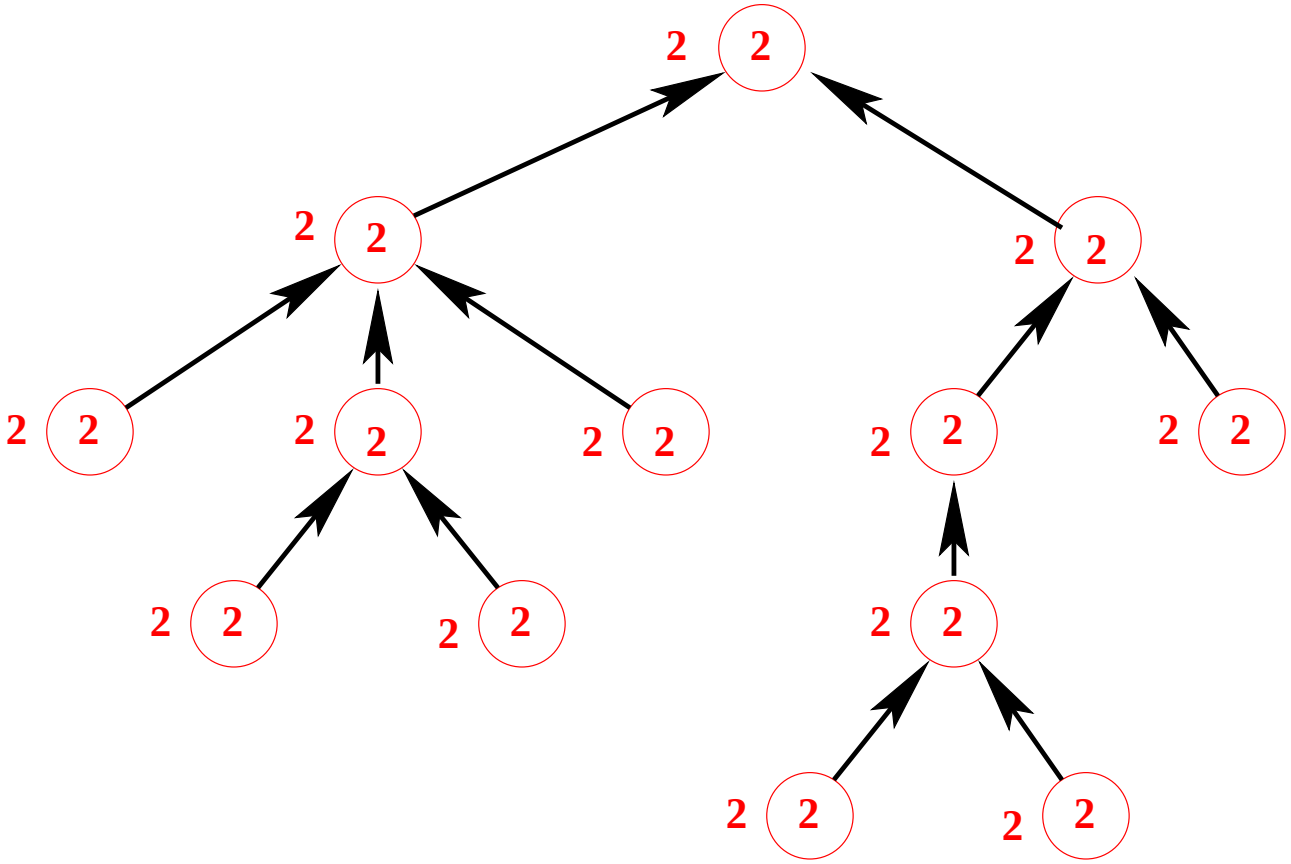
descending_2
when
$d(r) \neq c(r)$
then
$d(r) := d(r) + 1$
end

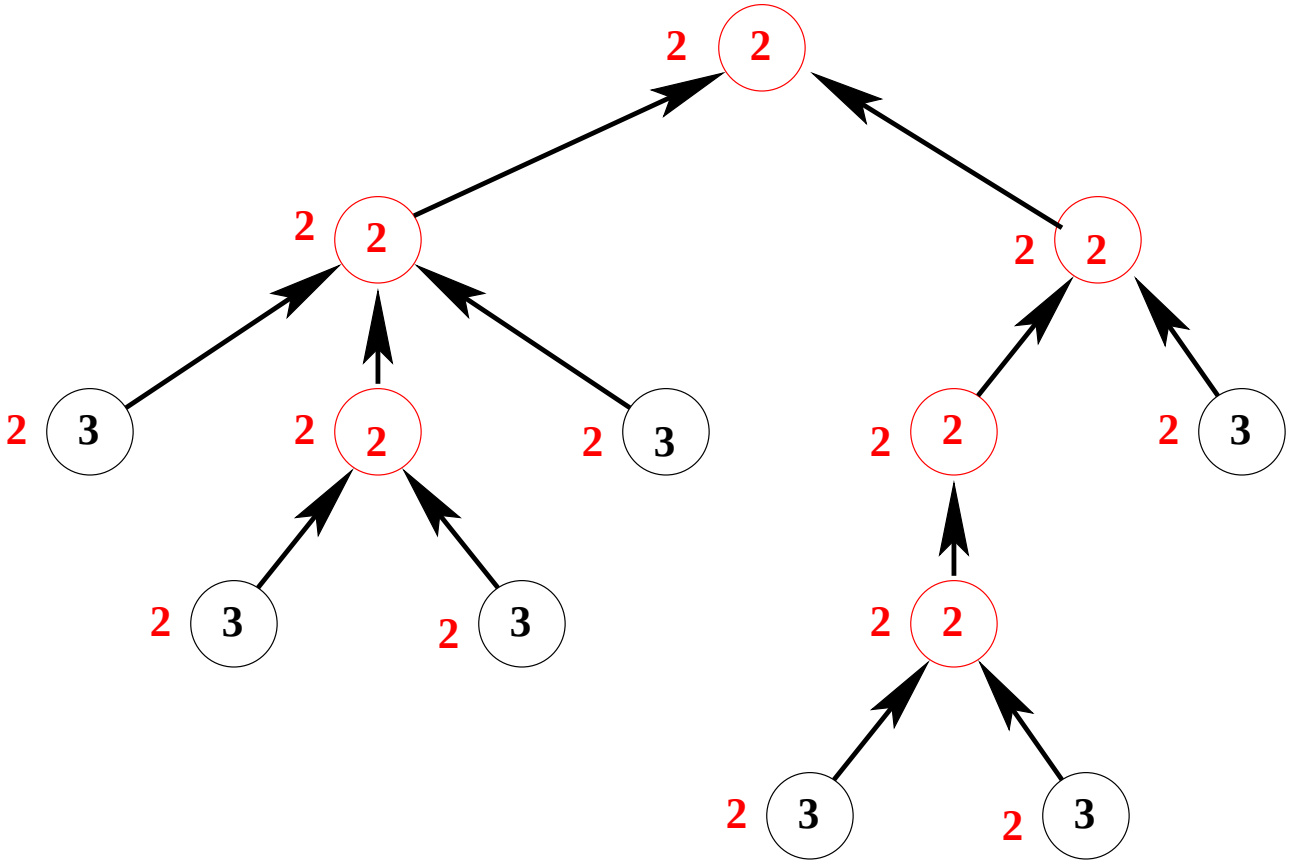

2 2
1 2
1 2
1 2 1 2 1 2
1 2 1 2
1 2 1 2

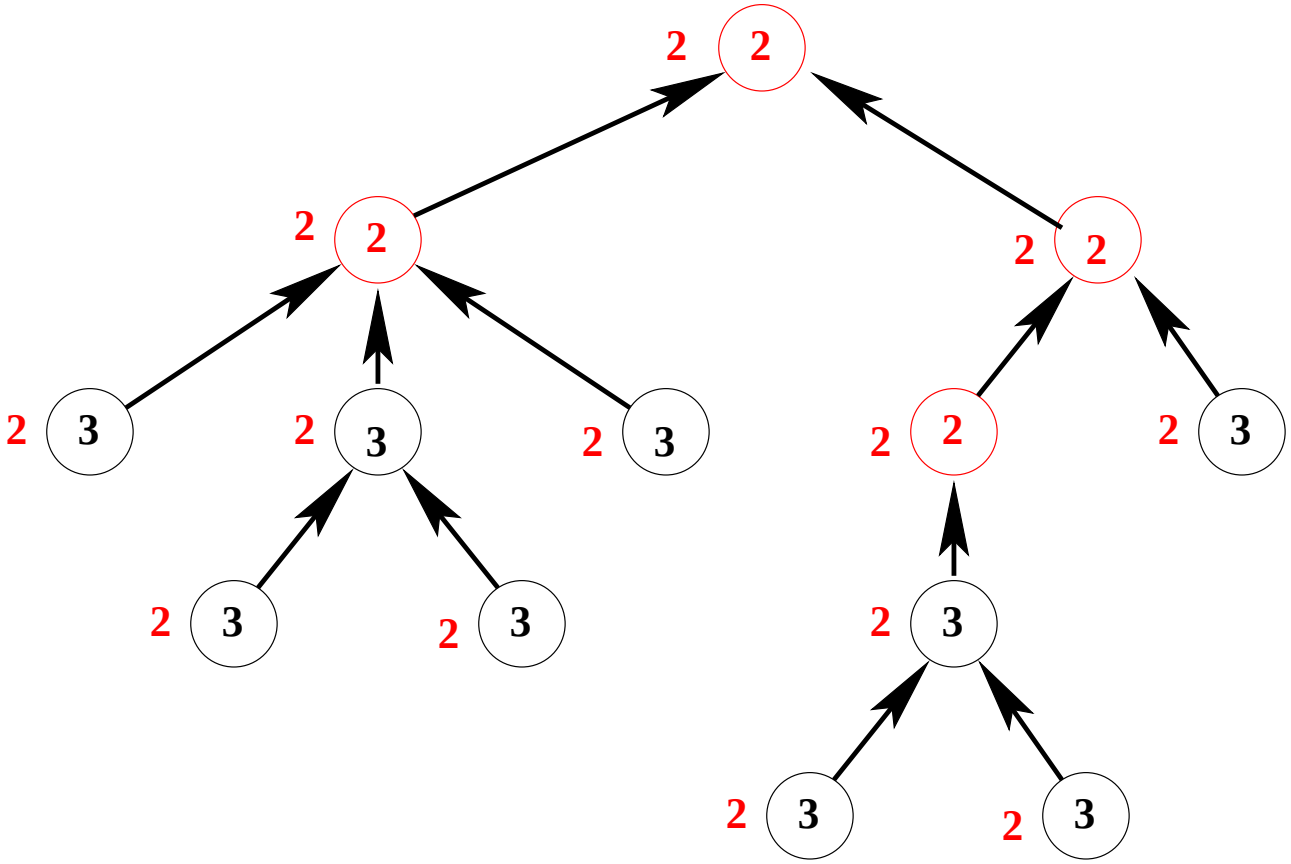

2 2
2 2
2 2
1 2
1 2
1 2
1 2
1 2
1 2
1 2
1 2

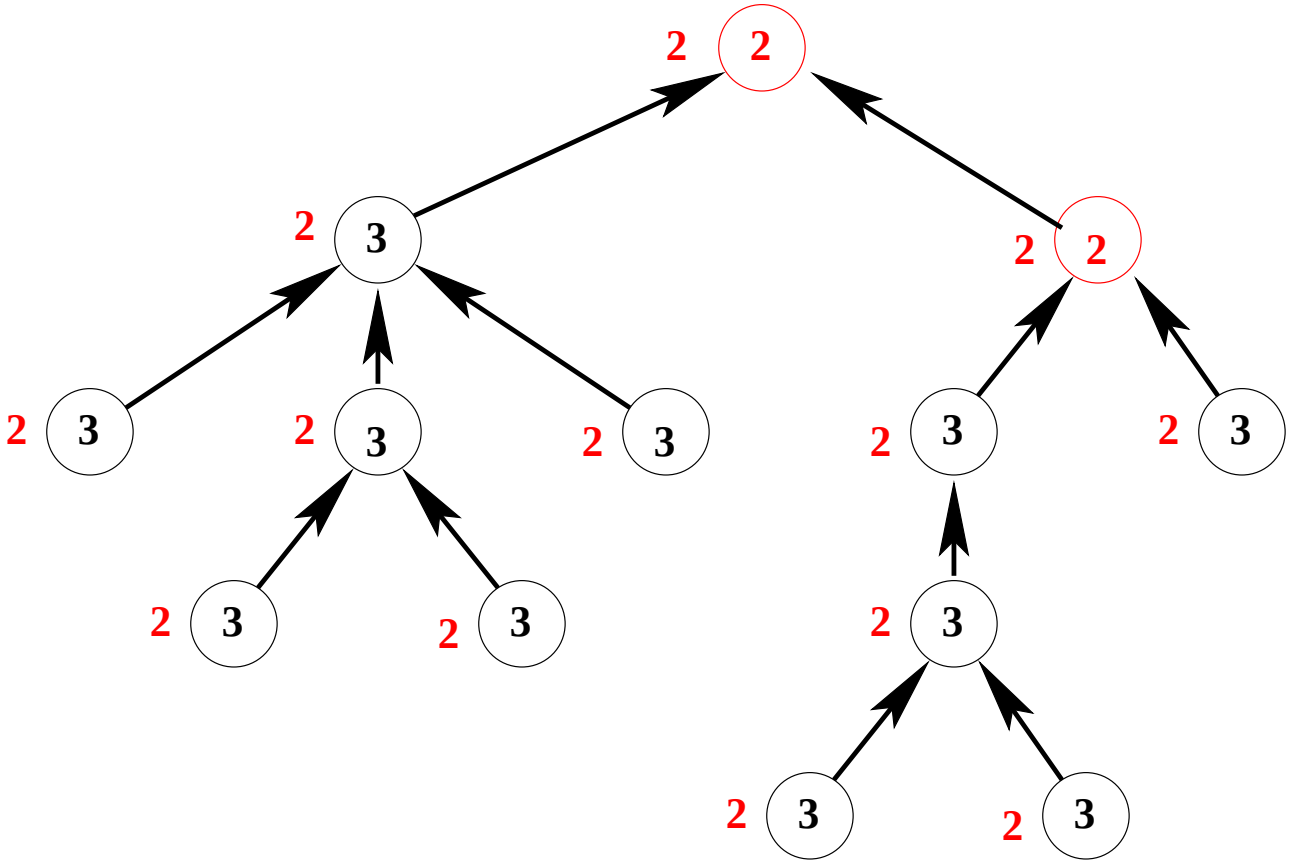

2 2
2 2
2 2
2 2
2 2
2 2
2 2
1 2
1 2
1 2
1 2

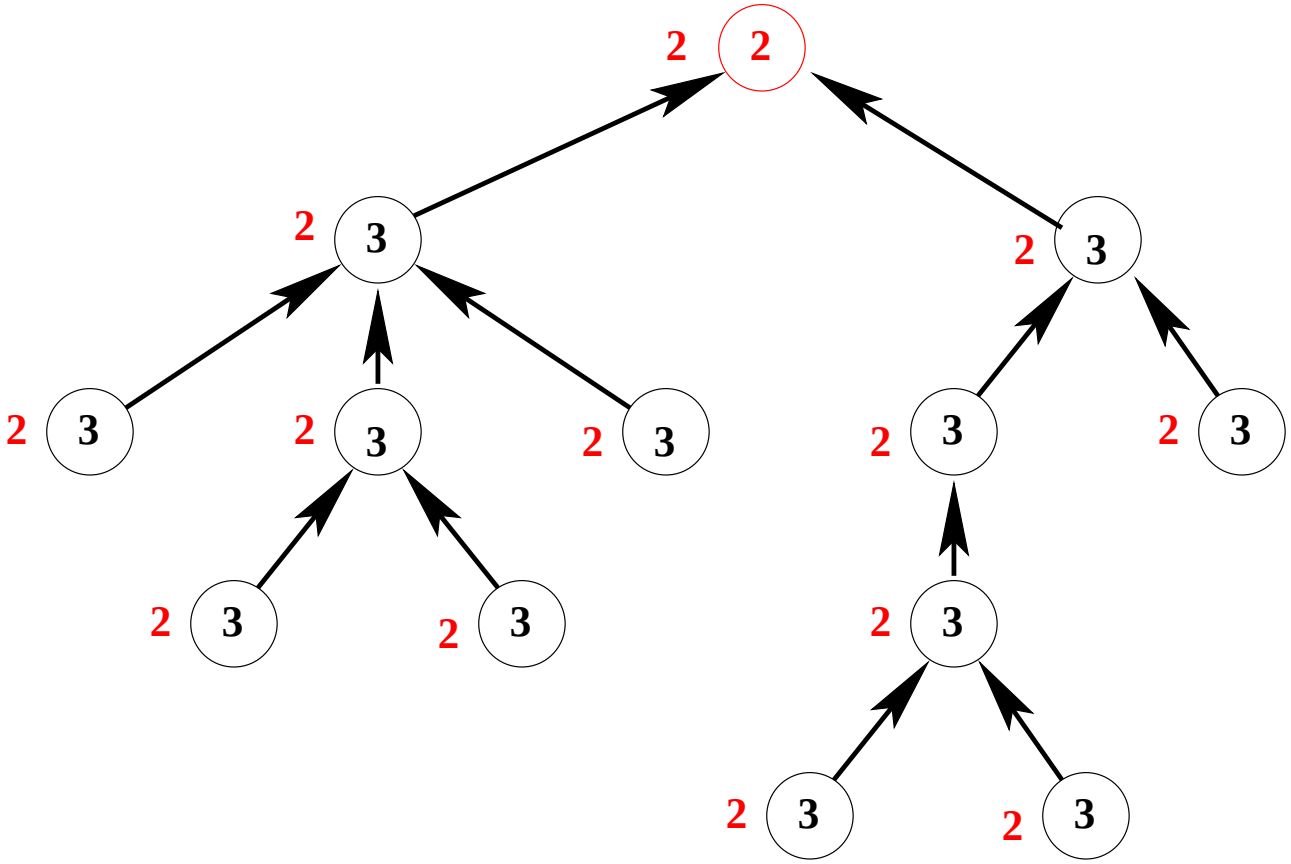

2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2
1 2
1 2

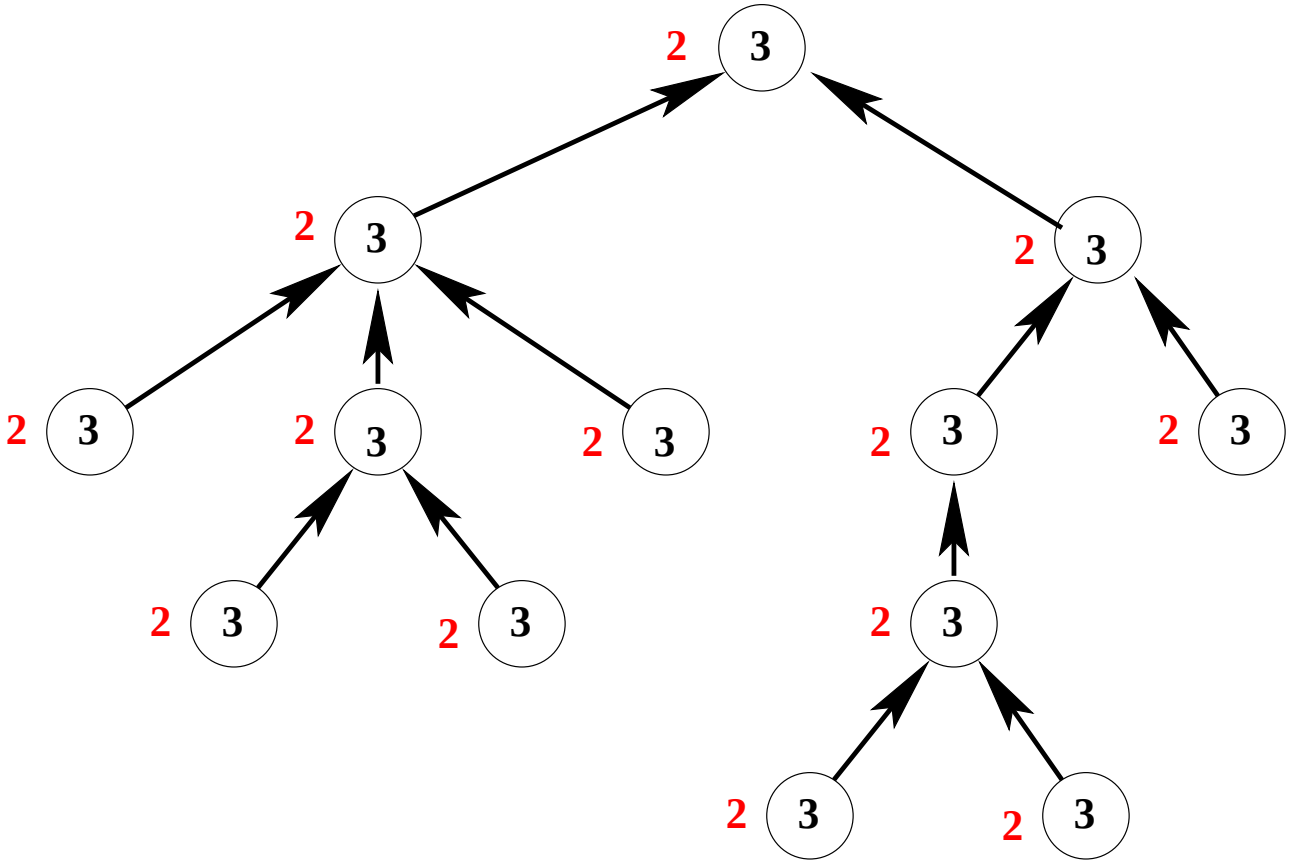

2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2
2 2


2 2
2 2
2 2
2 3
2 2
2 3
2 2
2 3
2 3
2 2
2 3
2 3


2 2
2 2
2 2
2 3
2 3
2 3
2 2
2 3
2 3
2 3
2 3
2 3


2 2
2 3
2 2
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3


2 2
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3


2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3
2 3

Fourth Refinement: the State

- We replace the counters by their parities

- we add the constant $parity$

carrier set: P

constants: $r, f, parity$

axm4_1: $parity \in \mathbb{N} \to \{0, 1\}$

axm4_2: $parity(0) = 0$

axm4_2: $\forall x \,.\, parity(x + 1) = 1 - parity(x)$

An Important Theorem about Parities

thm4_1: $\forall\, x, y \,.\, x \in y \,..\, y + 1 \Rightarrow (parity(x) = parity(y) \;\Leftrightarrow\; x = y)$

Fourth Refinement: the State

- We replace c and d by p and q

variables: p, q

inv4_1: $p \in P \rightarrow \{0, 1\}$

inv4_2: $q \in P \rightarrow \{0, 1\}$

inv4_3: $\forall n \,.\, p(n) = parity(c(n))$

inv4_4: $\forall n \,.\, q(n) = parity(d(n))$

Fourth Refinement: the Events

ascending
any n **where**
$n \in P$
$p(n) = q(n)$
$\forall m \cdot (m \in f^{-1}[\{n\}] \Rightarrow p(m) \neq p(n))$
then
$p(n) := 1 - p(n)$
end

descending_1
any n **where**
$n \in P \setminus \{r\}$
$q(n) \neq q(f(n))$
then
$q(n) := 1 - q(n)$
end

descending_2
when
$p(r) \neq q(r)$
then
$q(r) := 1 - q(r)$
end