Some Generalised Reductions of Ordered Binary Decision Diagramm (GroBdd)

Joan Thibault

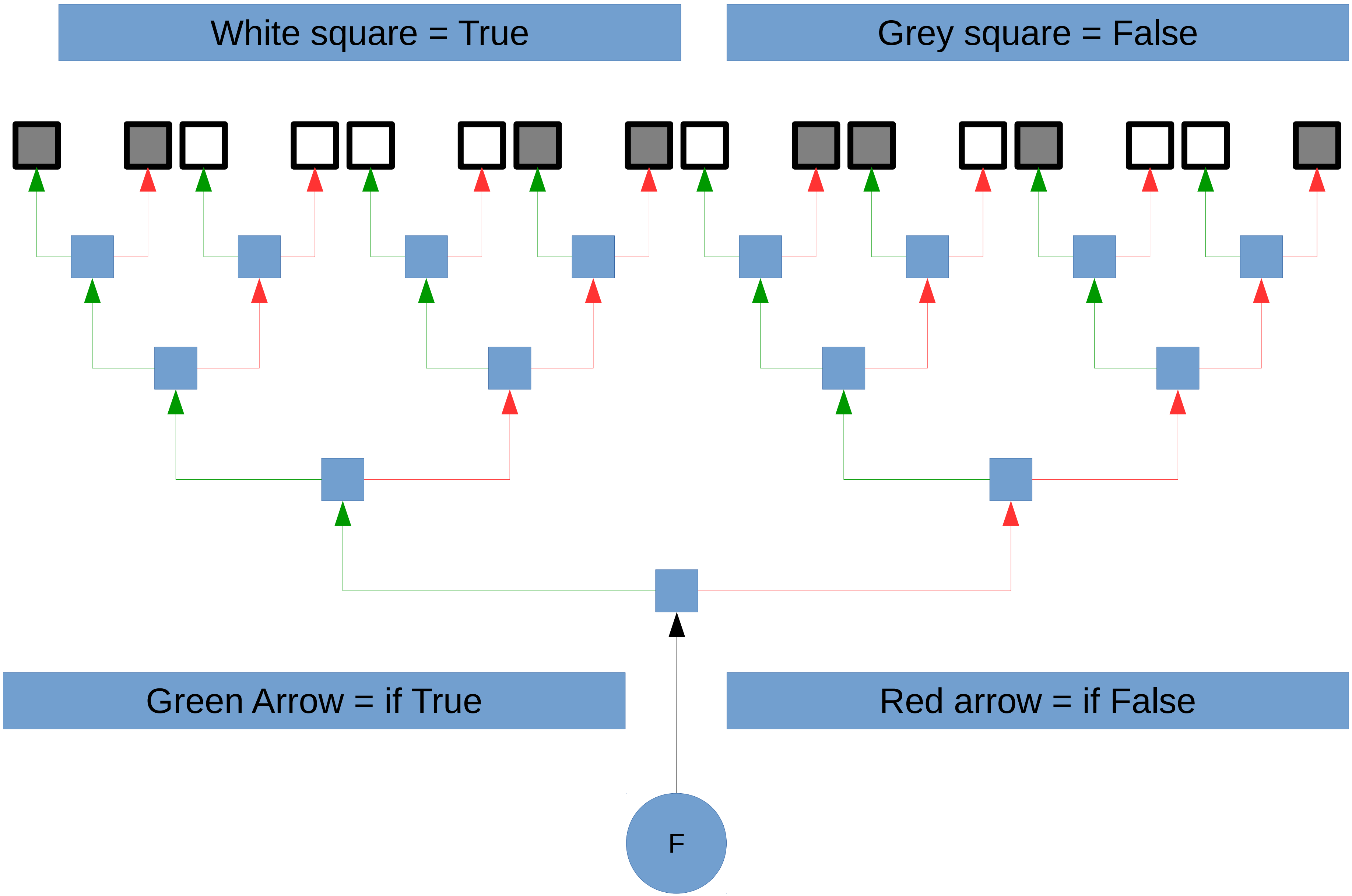
Boolean Functions

- Why ?
  - Computer Aided Design (e.g. digital circuit synthesis)
  - Knowledge Representation (e.g. Artificial Intelligence)
  - Combinatorial Problems (e.g. N-Queens problem)
- What ?
  - Compact representation
  - Operations (e.g. composing, concatening, evaluation)
  - Operators (e.g. AND, XOR, ITE, NOT)
  - Reductions (e.g. quantification, partial evaluation, SAT)

Boolean Functions

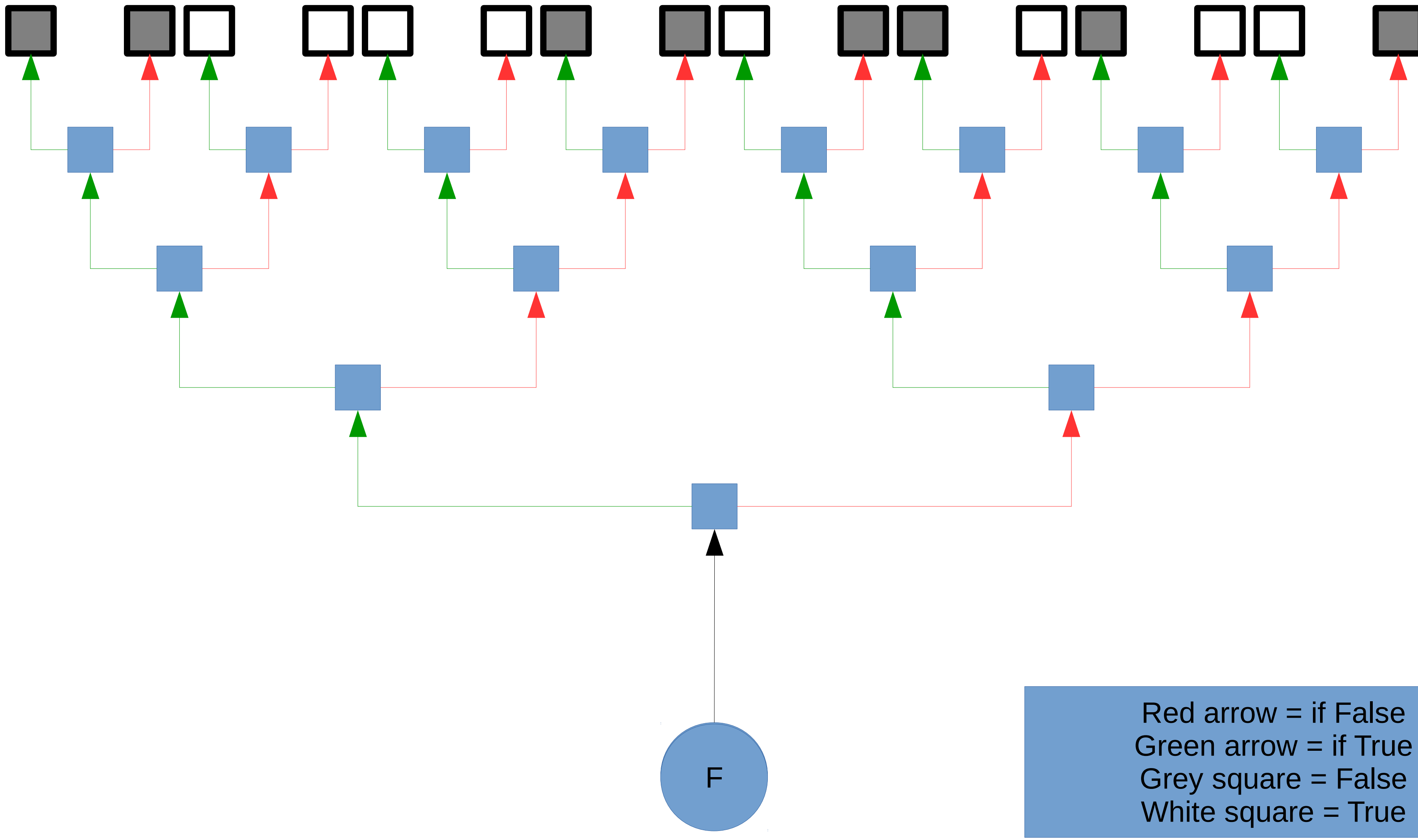
- Various representations
  - Truth Table
  - Conjonctive / Disjonctive Normal Form
  - And Inverter Graph
  - Binary Decision Diagramm
    - Reduced Ordered BDD
    - Zero supressed BDD
    - Xor based BDD

Section 1
What is a ROBDD ?


White square = True
Grey square = False
Green Arrow = if True
Red arrow = if False
F

(Bryant) Step 1: we merge isomorphic sub-graphs

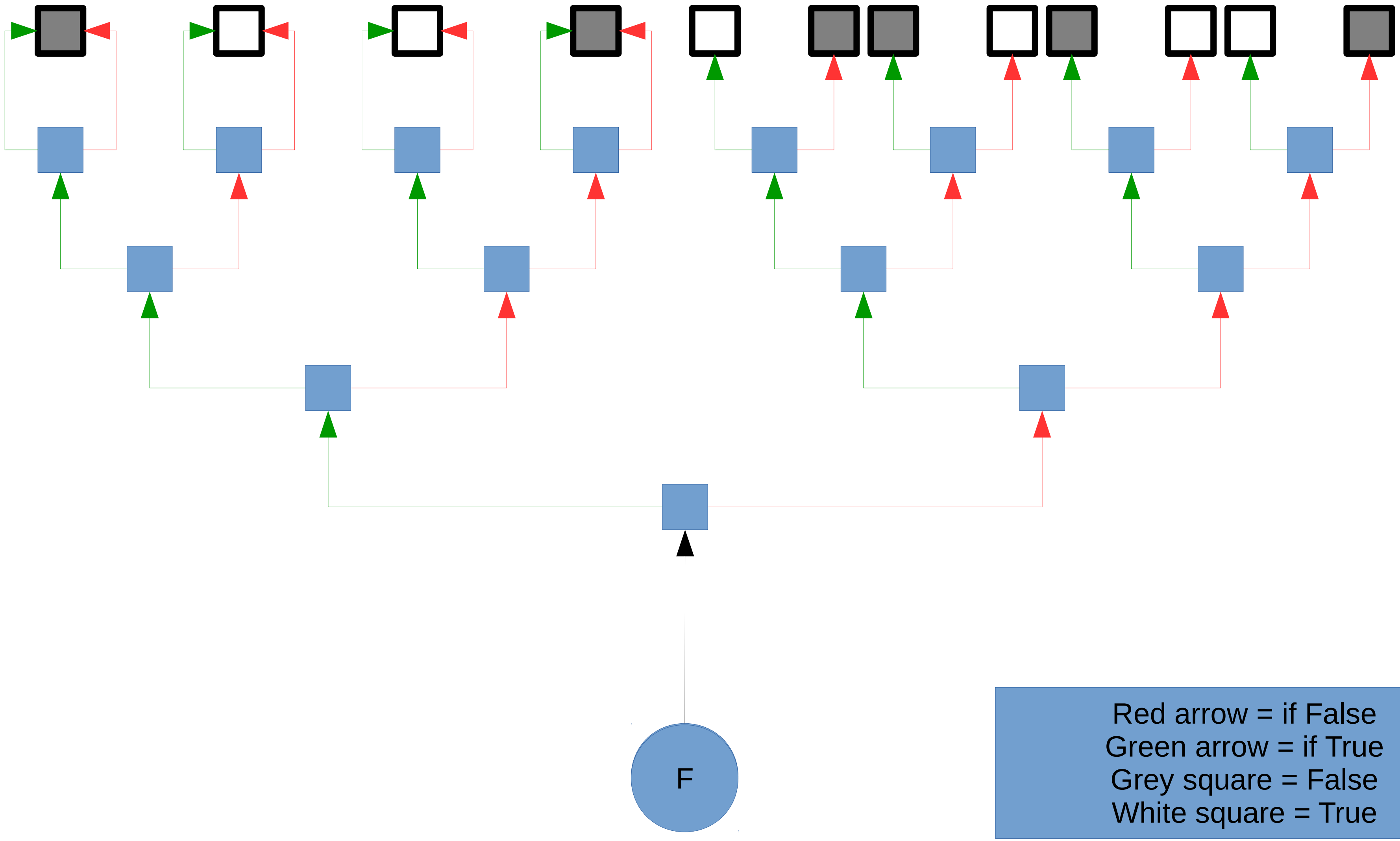
For aesthetic reasons, we do not merge terminals



Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Bryant) Step 1: we merge isomorphic sub-graphs

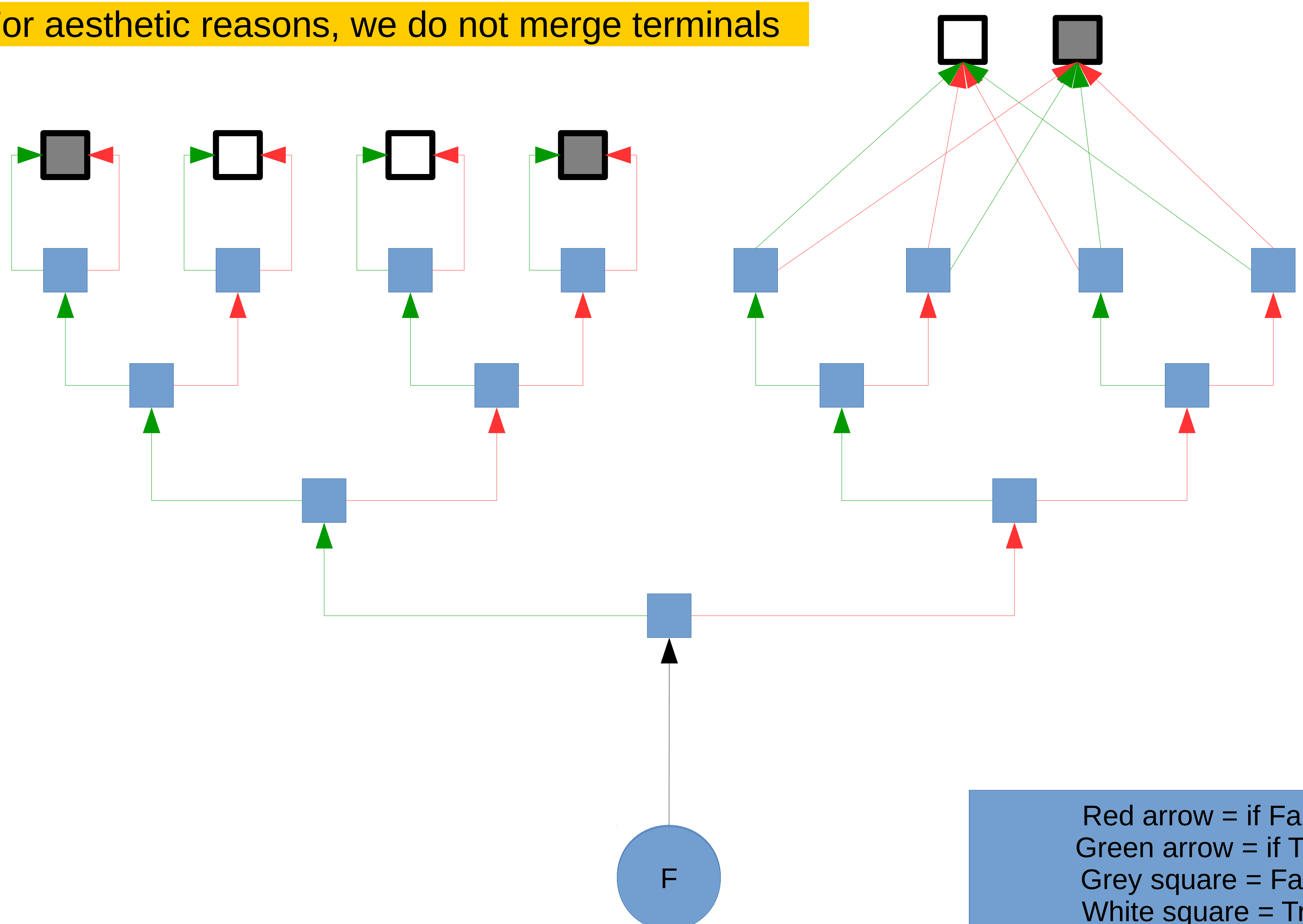
For aesthetic reasons, we do not merge terminals



Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

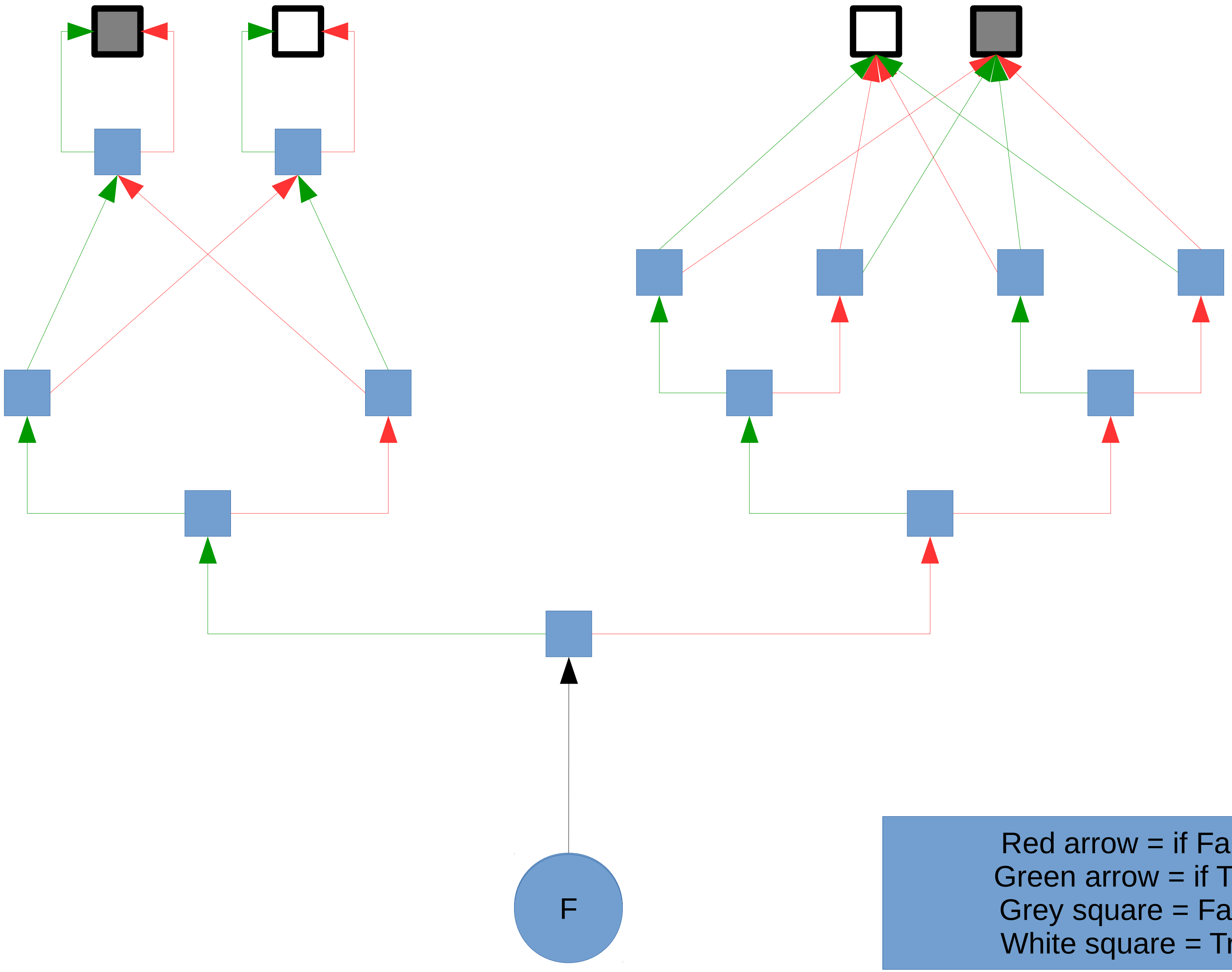
(Bryant) Step 1: we merge isomorphic sub-graphs

For aesthetic reasons, we do not merge terminals



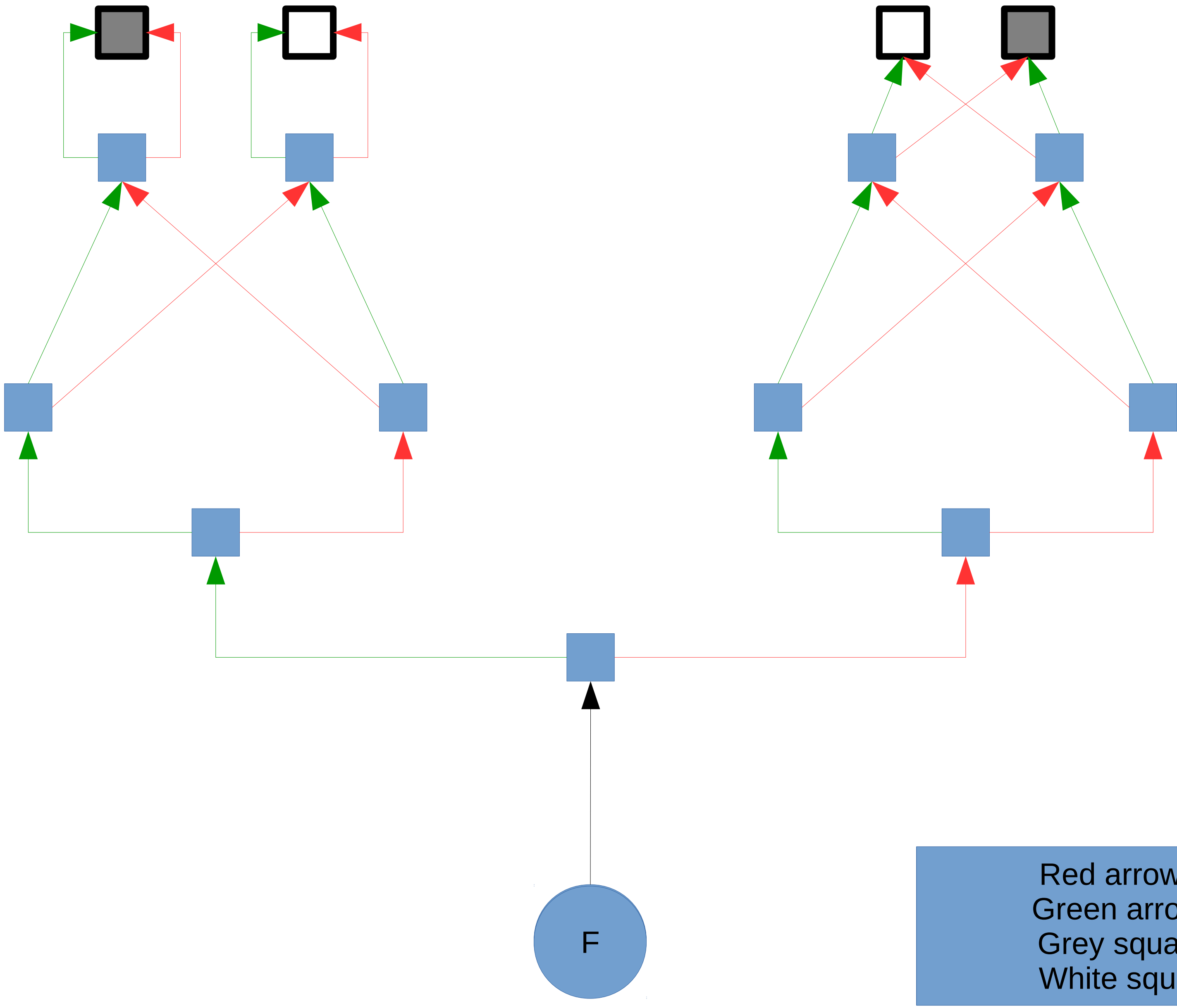
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Bryant) Step 1: we merge isomorphic sub-graphs



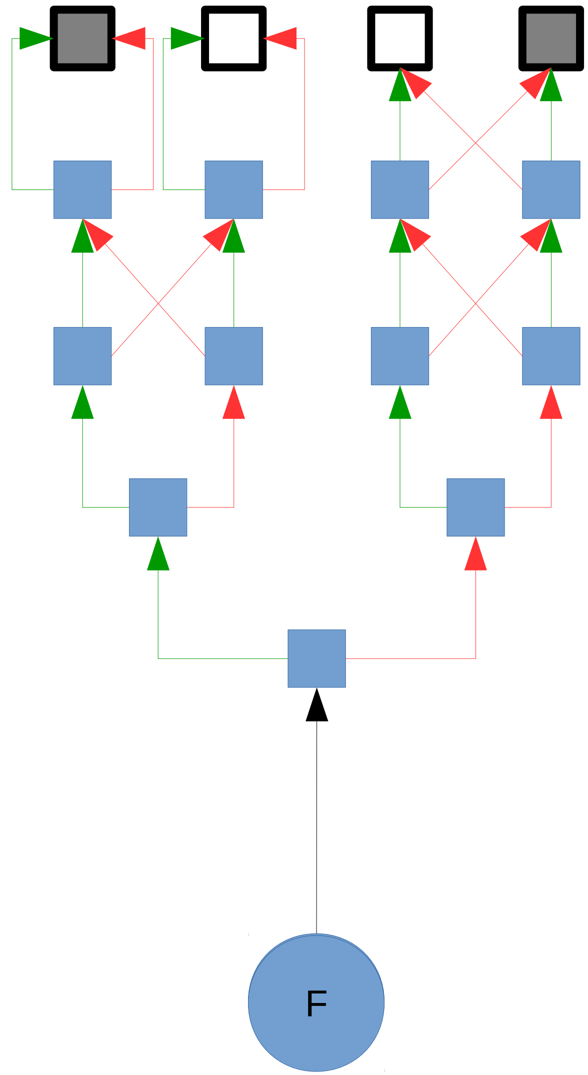
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Bryant) Step 1: we merge isomorphic sub-graphs



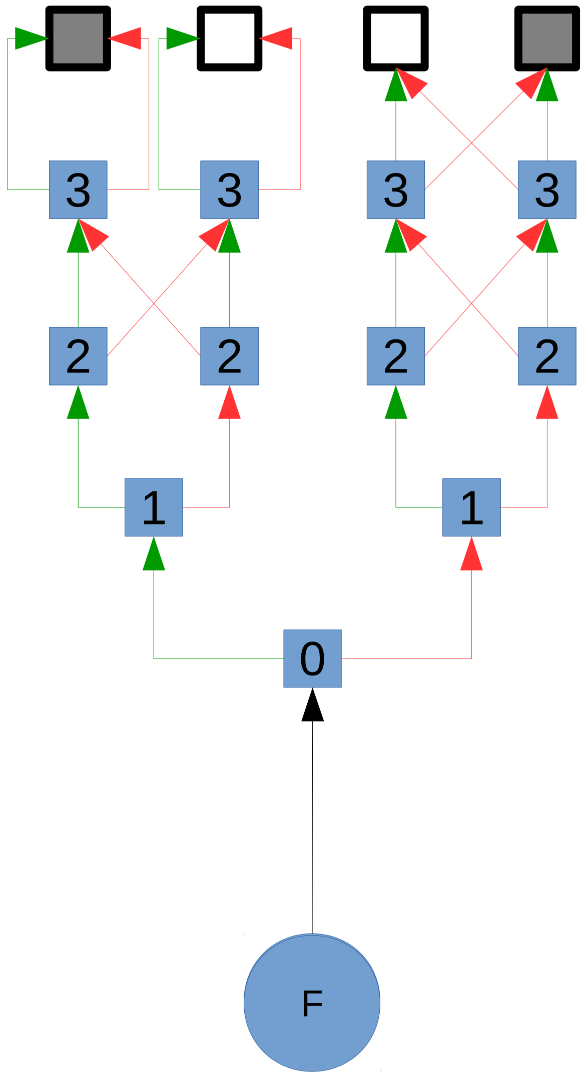
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Bryant) Step 1: we merge isomorphic sub-graphs



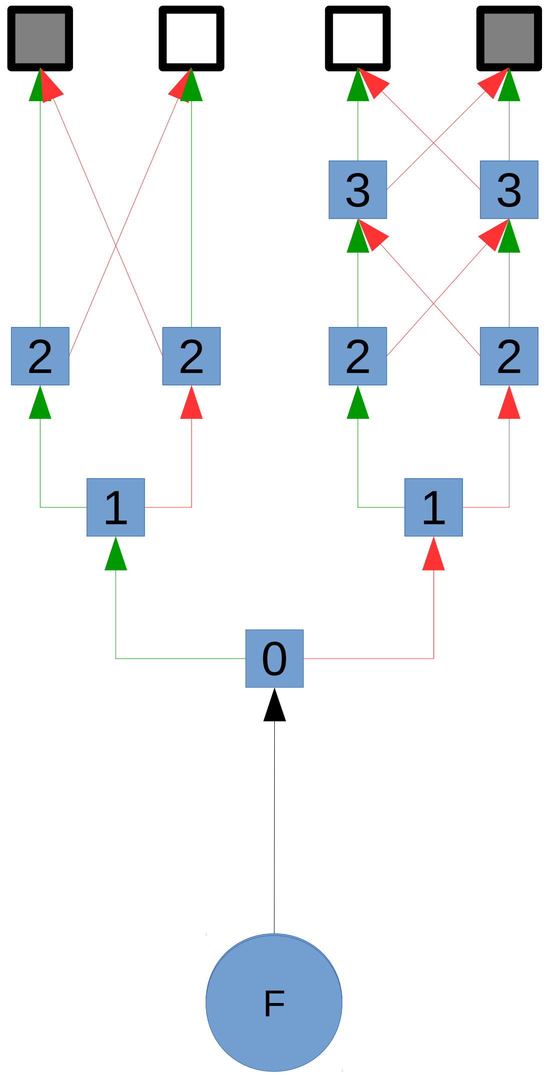
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Bryant) Step 2: we specify the depth of each node



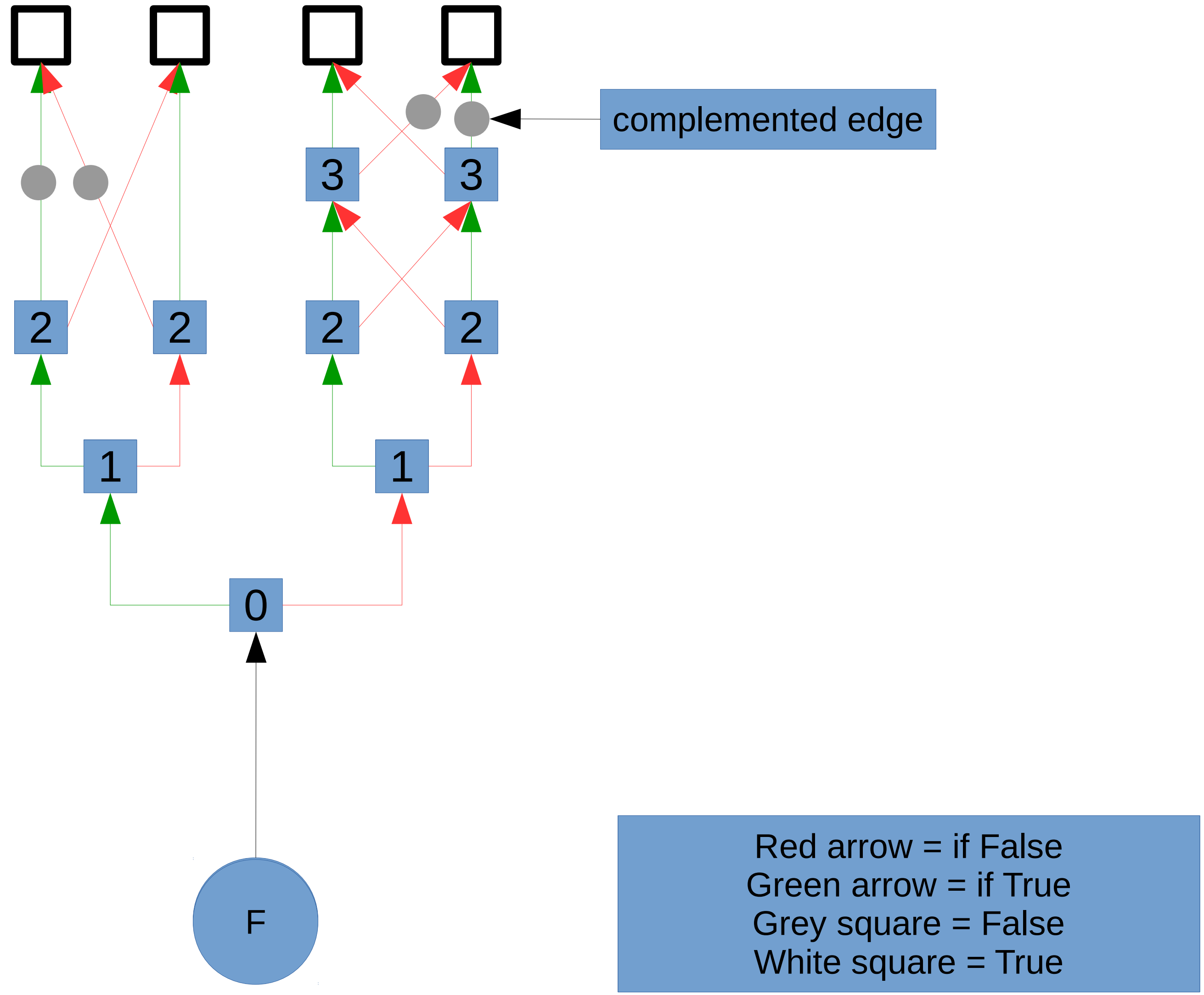
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Bryant) Step 3: we remove useless decisions

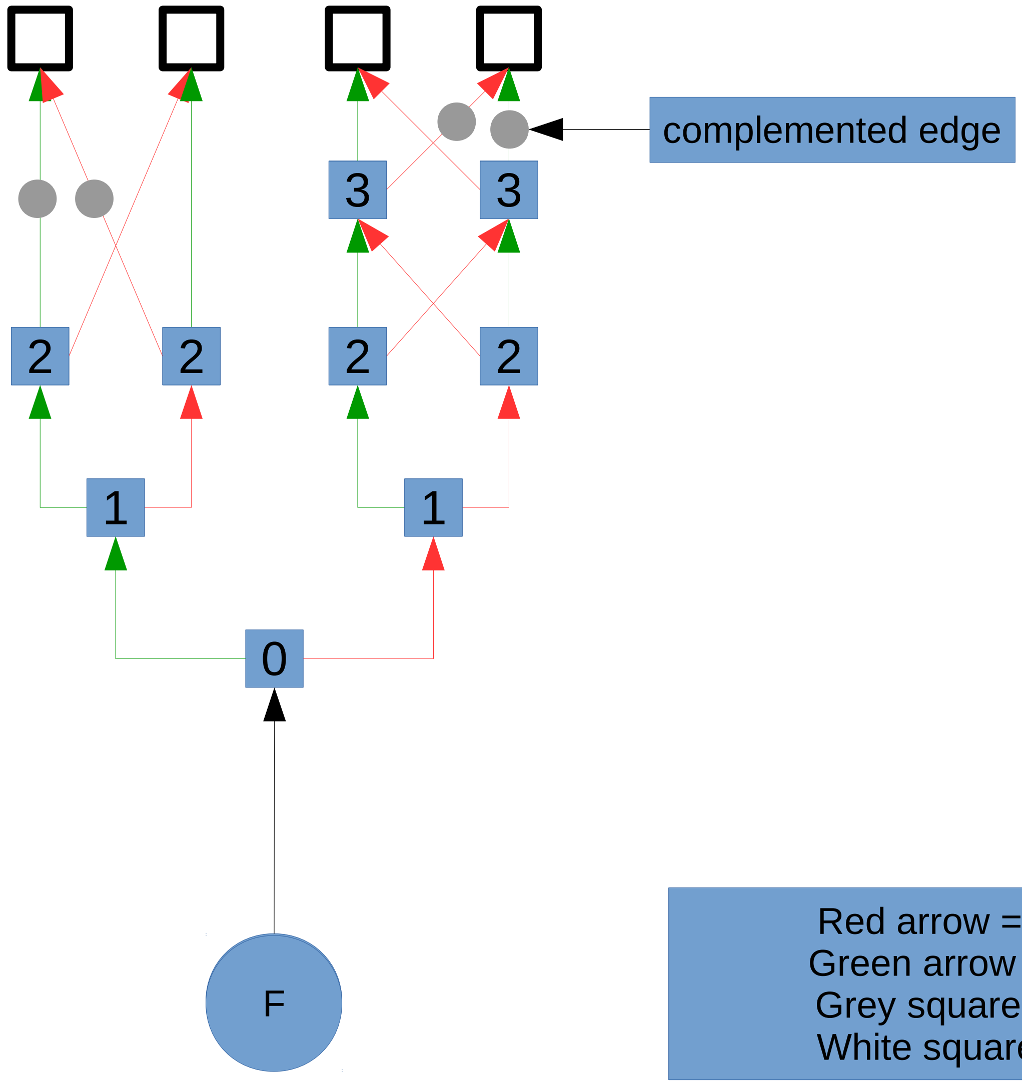


Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Complemented Edges) Step 1: we replace the False node by a complemented edge to True

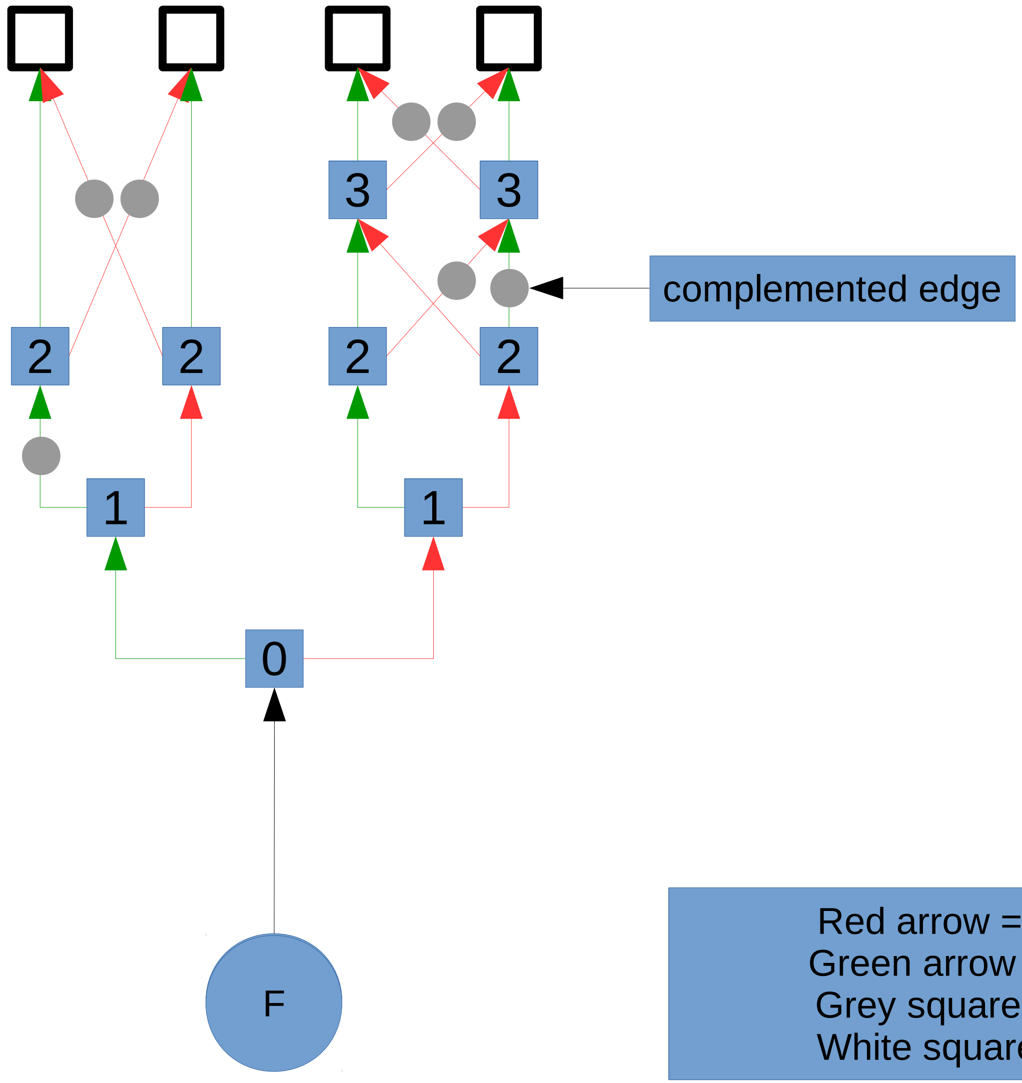


(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented



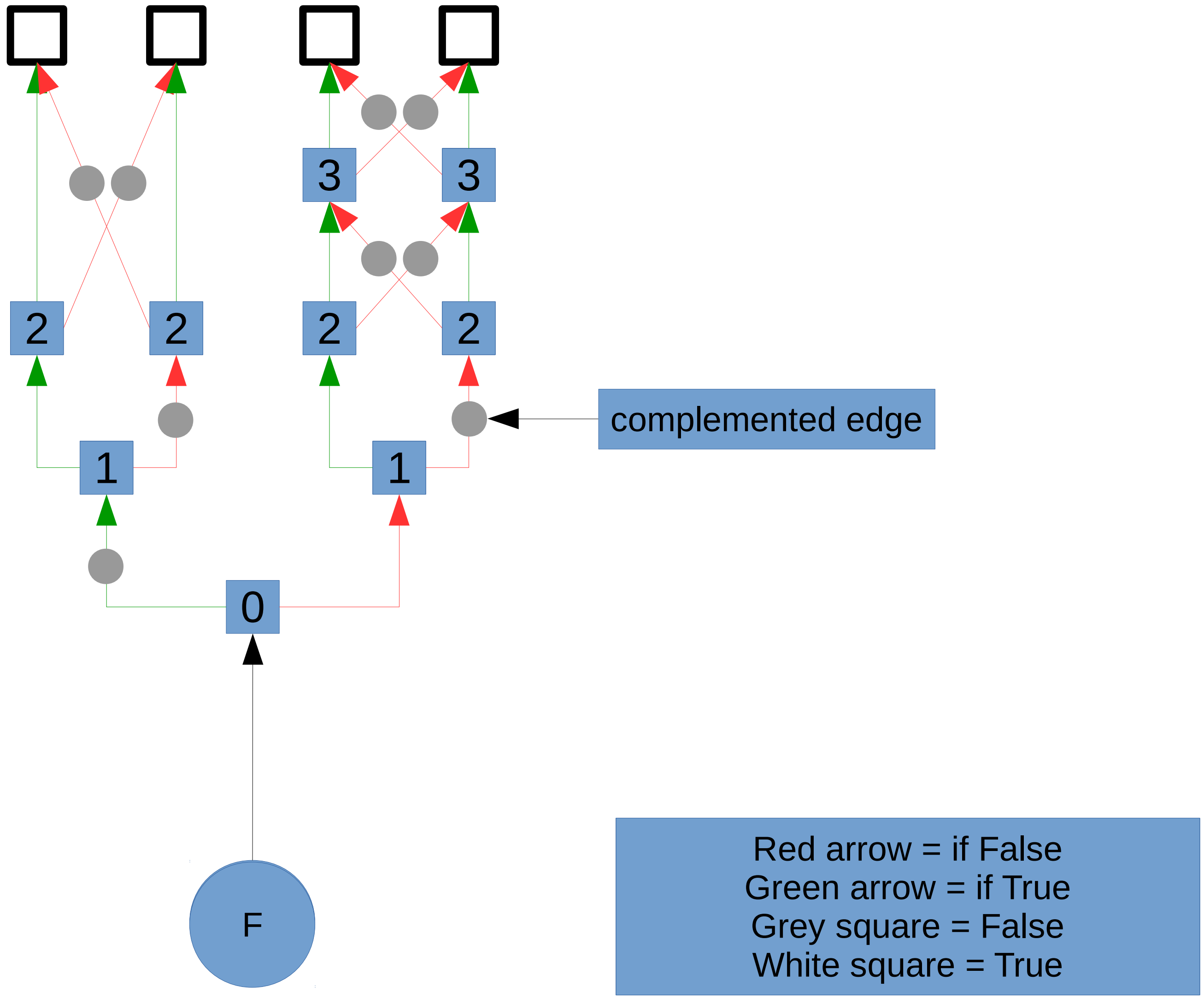
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented

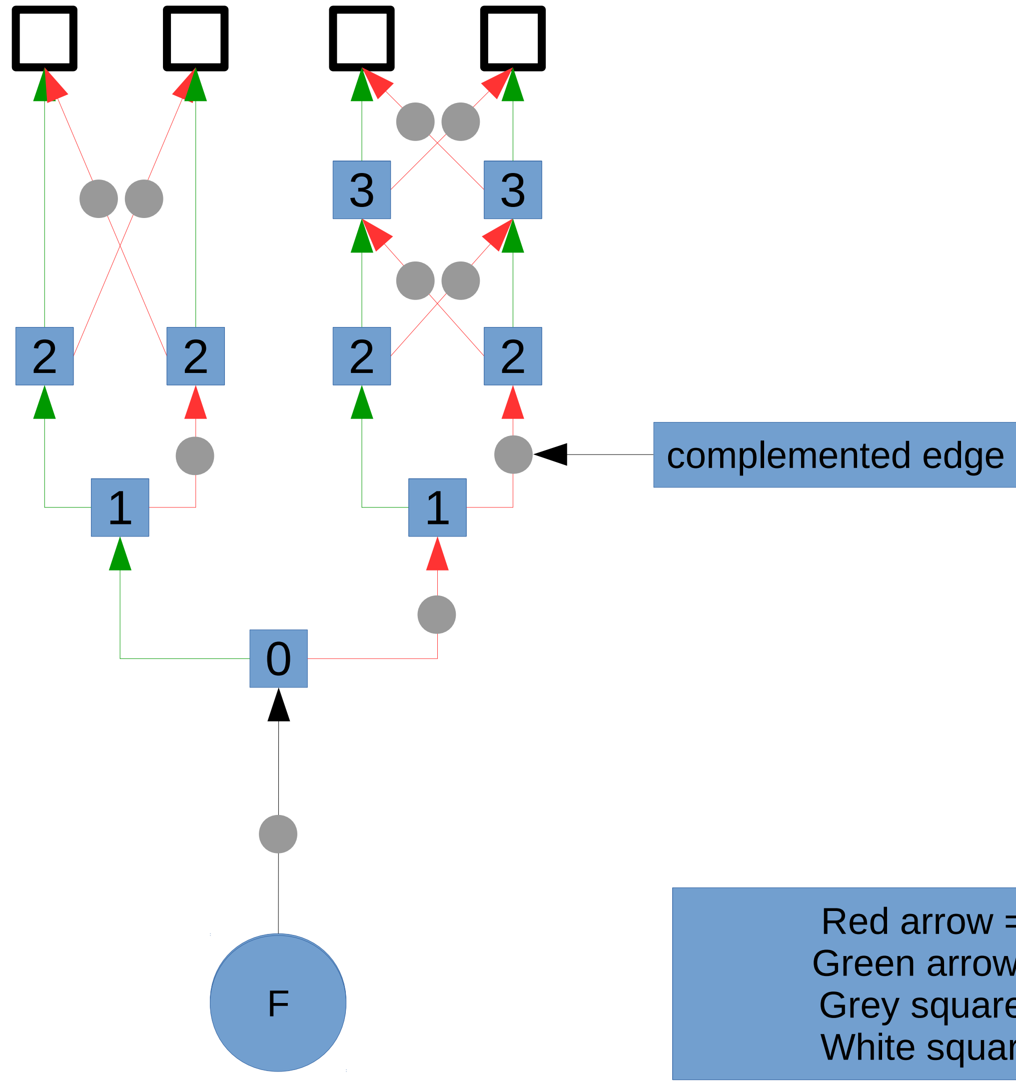


Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented

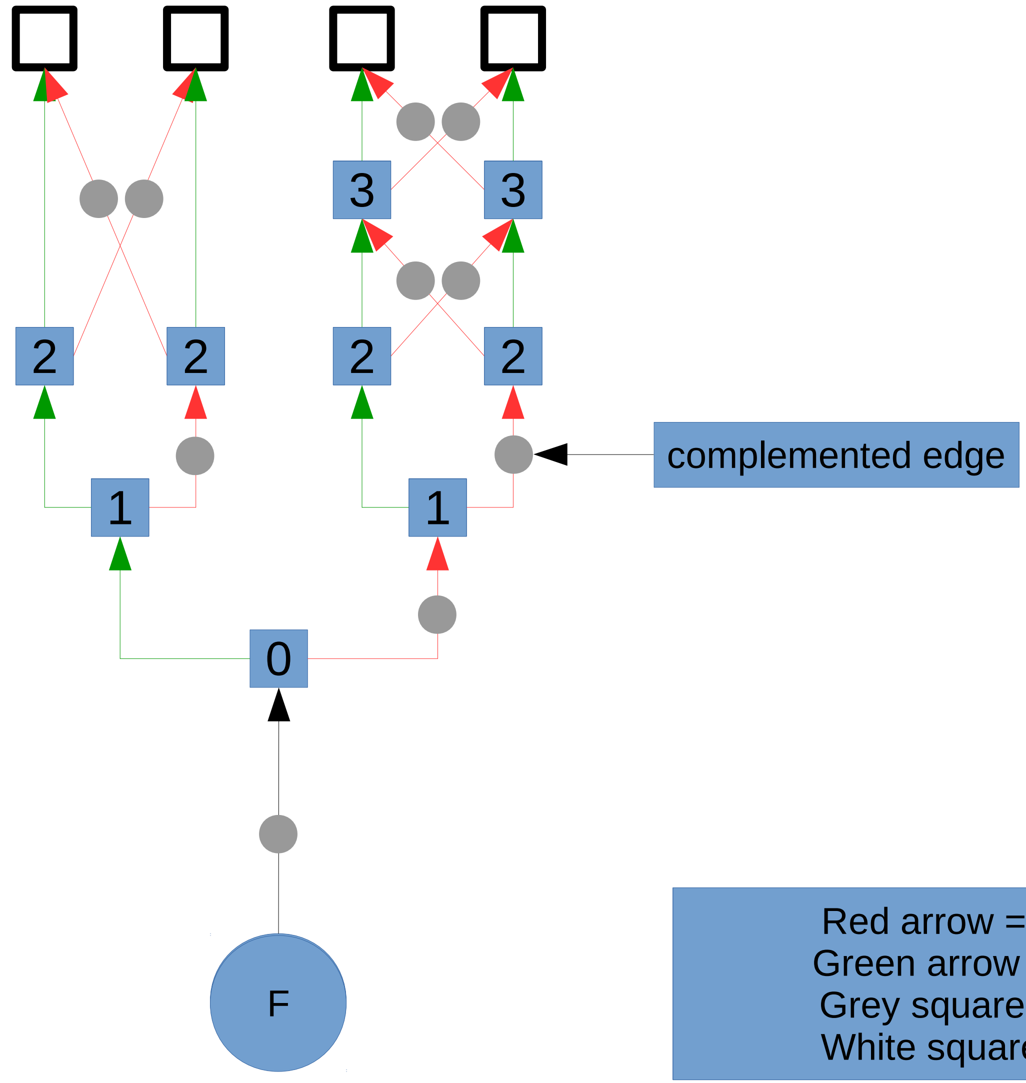


(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented



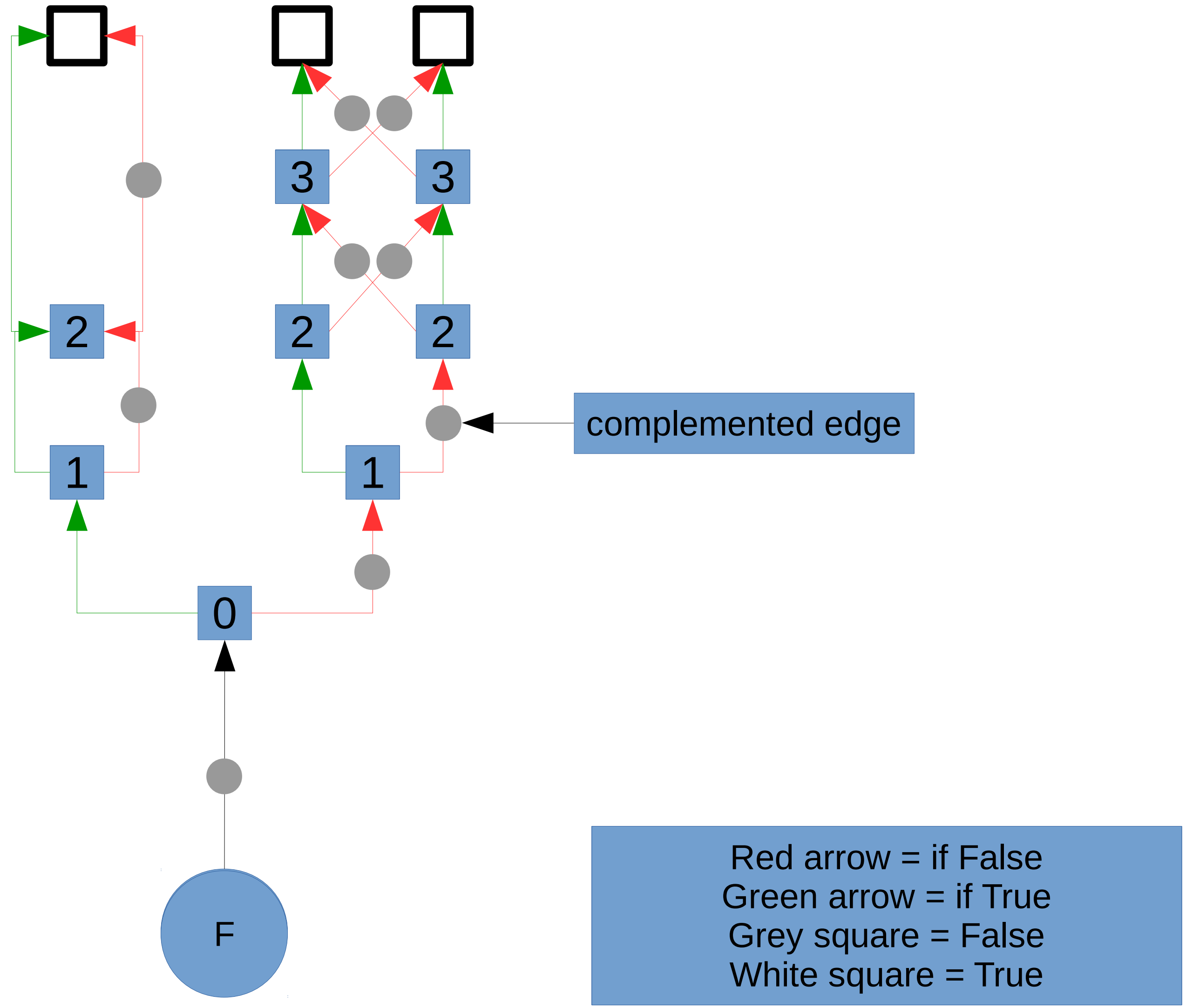
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

we merge isomorphic sub-graphs

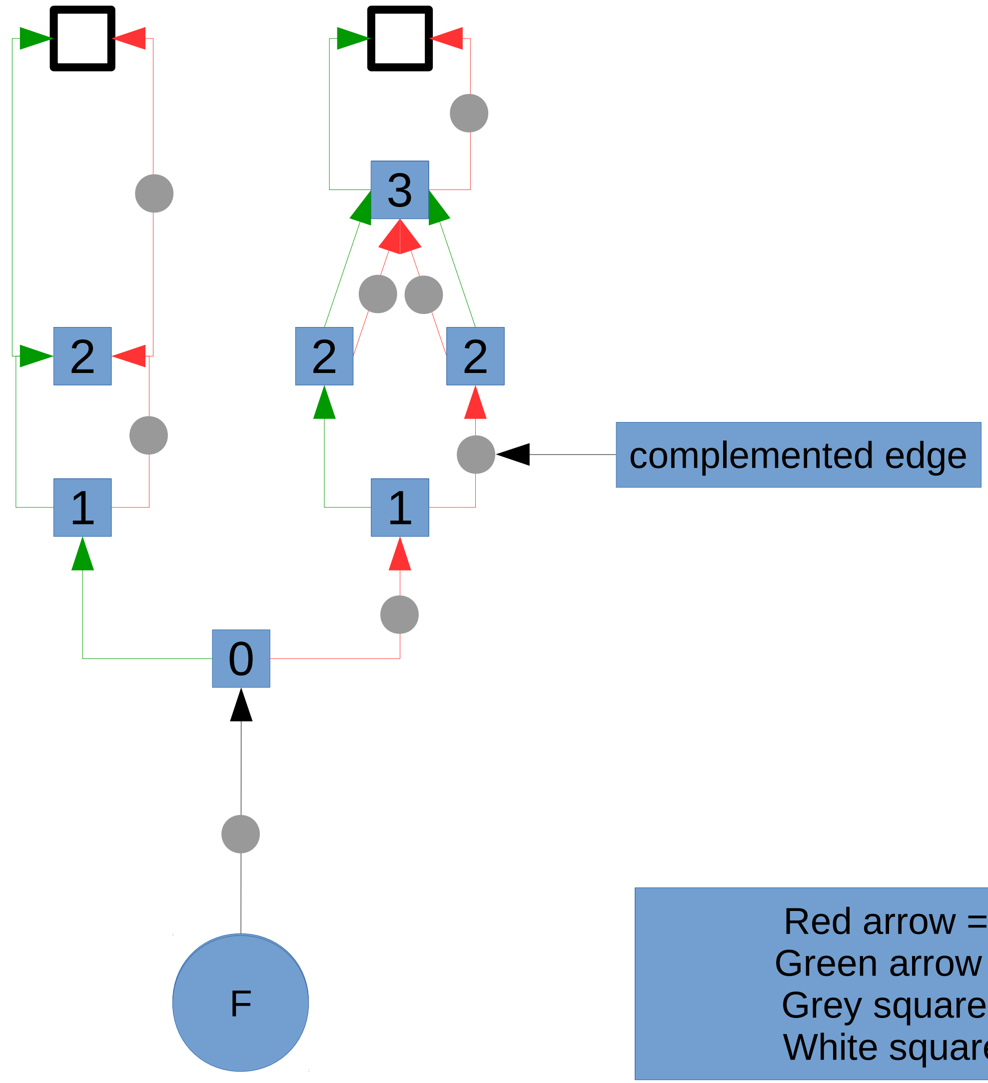


Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

we merge isomorphic sub-graphs

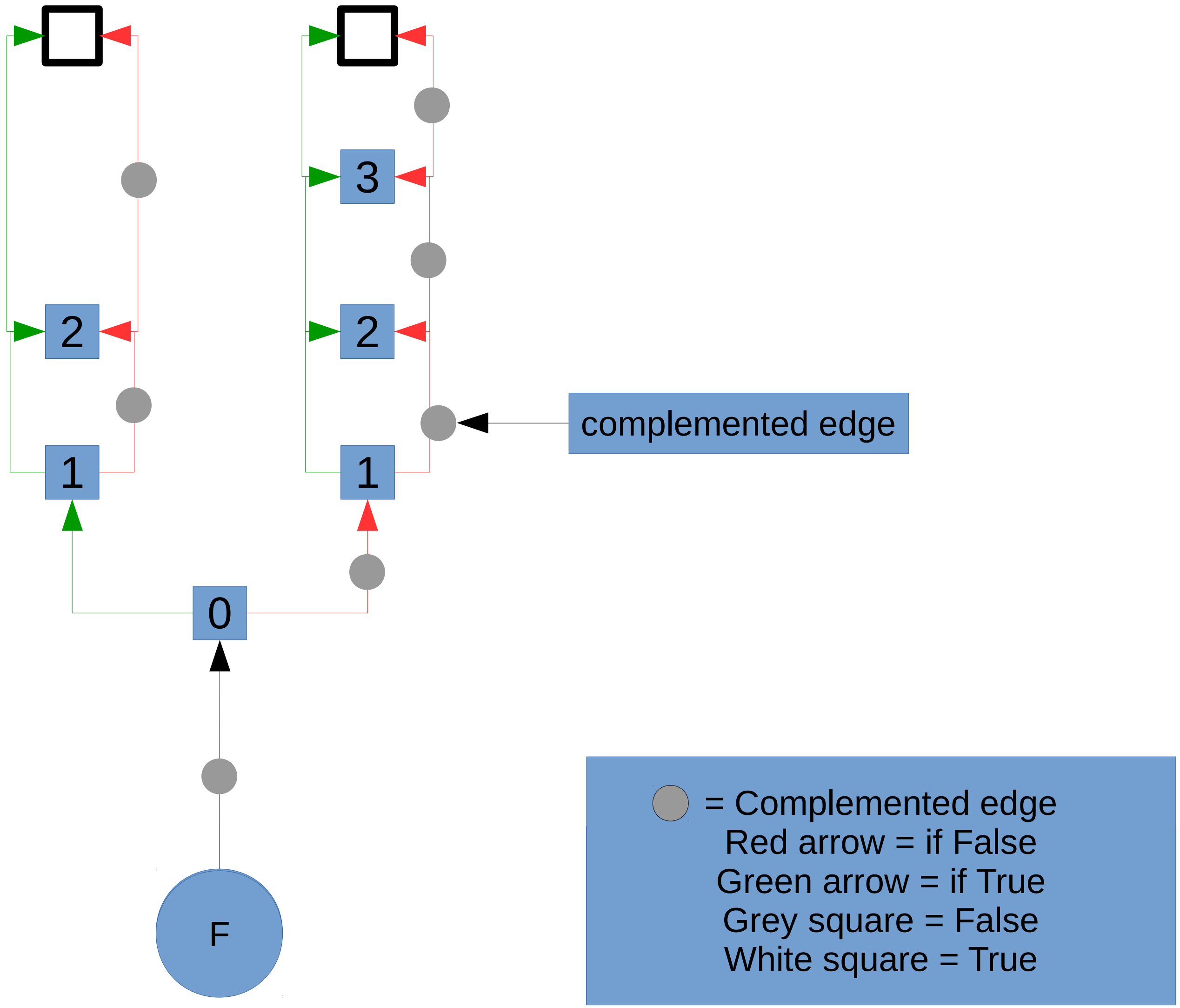


we merge isomorphic sub-graphs



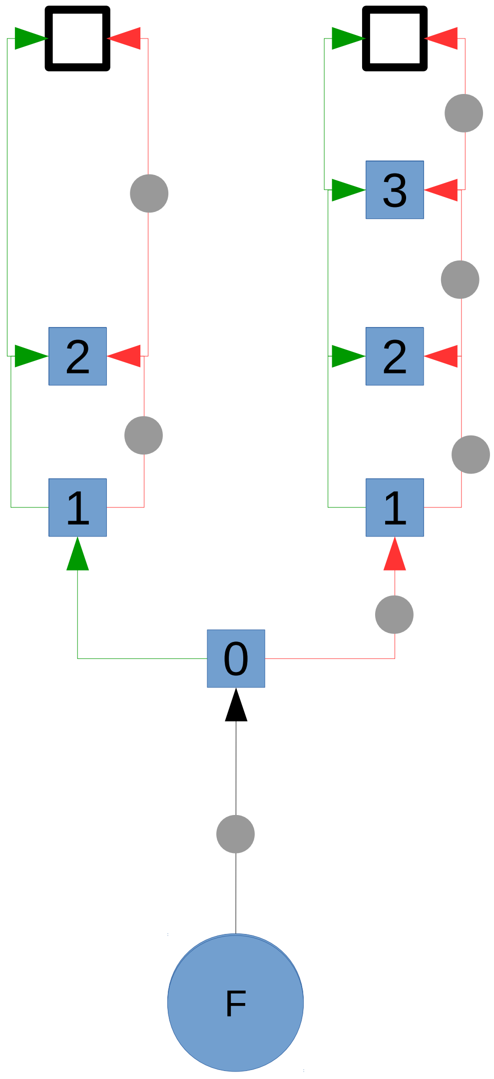
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

we merge isomorphic sub-graphs



Augmenting edges with complementation can be performed in linear time in #node

State Of The Art since 2000s

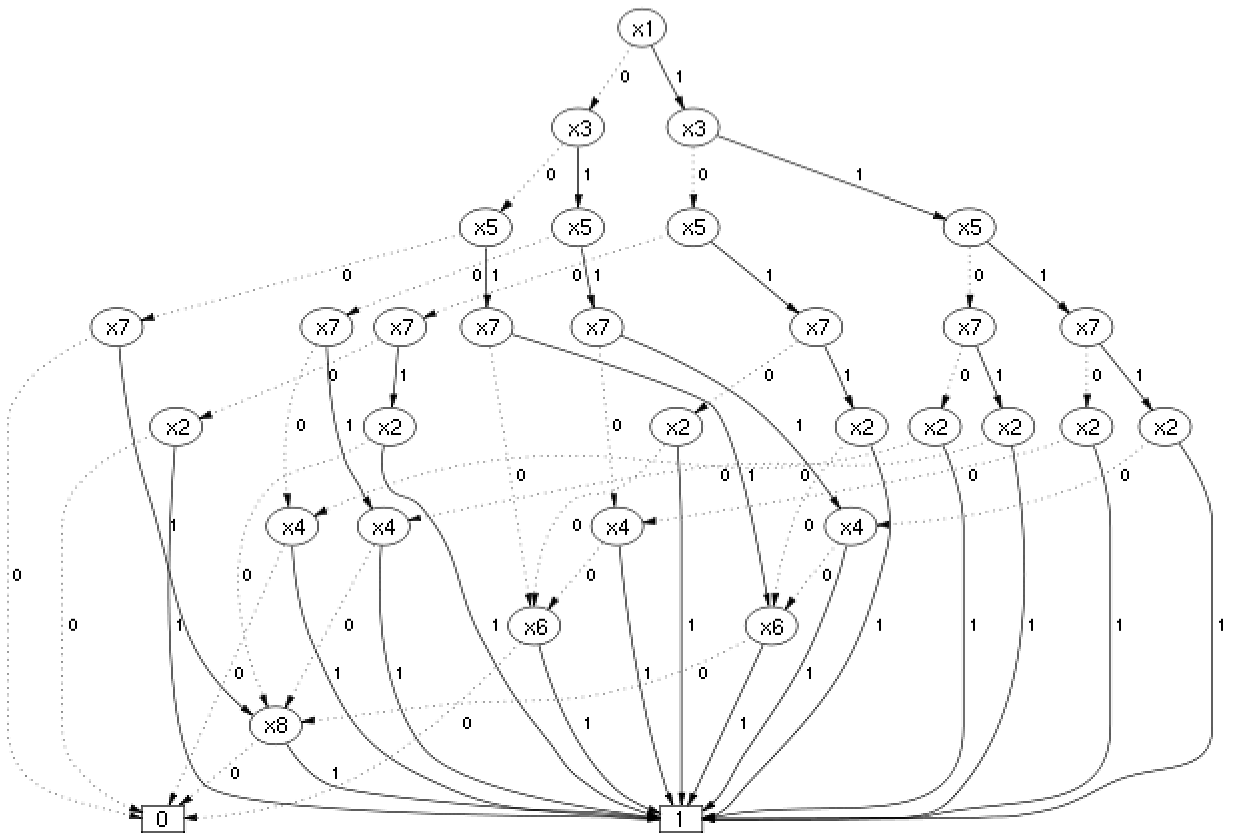


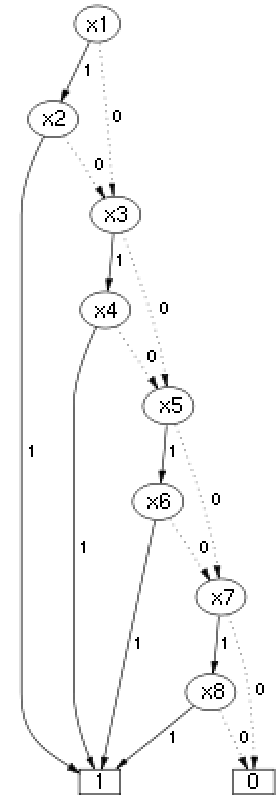
= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

Reduced Ordered BDD

- =) :
  - SAT : constant time
  - Any/Max/Min SAT : linear time (#variable)
  - #SAT : linear time (#node)
  - NOT : constant time
- =(:
  - AND, XOR : quadratic time/space (#node)
  - #node is order dependent

#node is order dependent

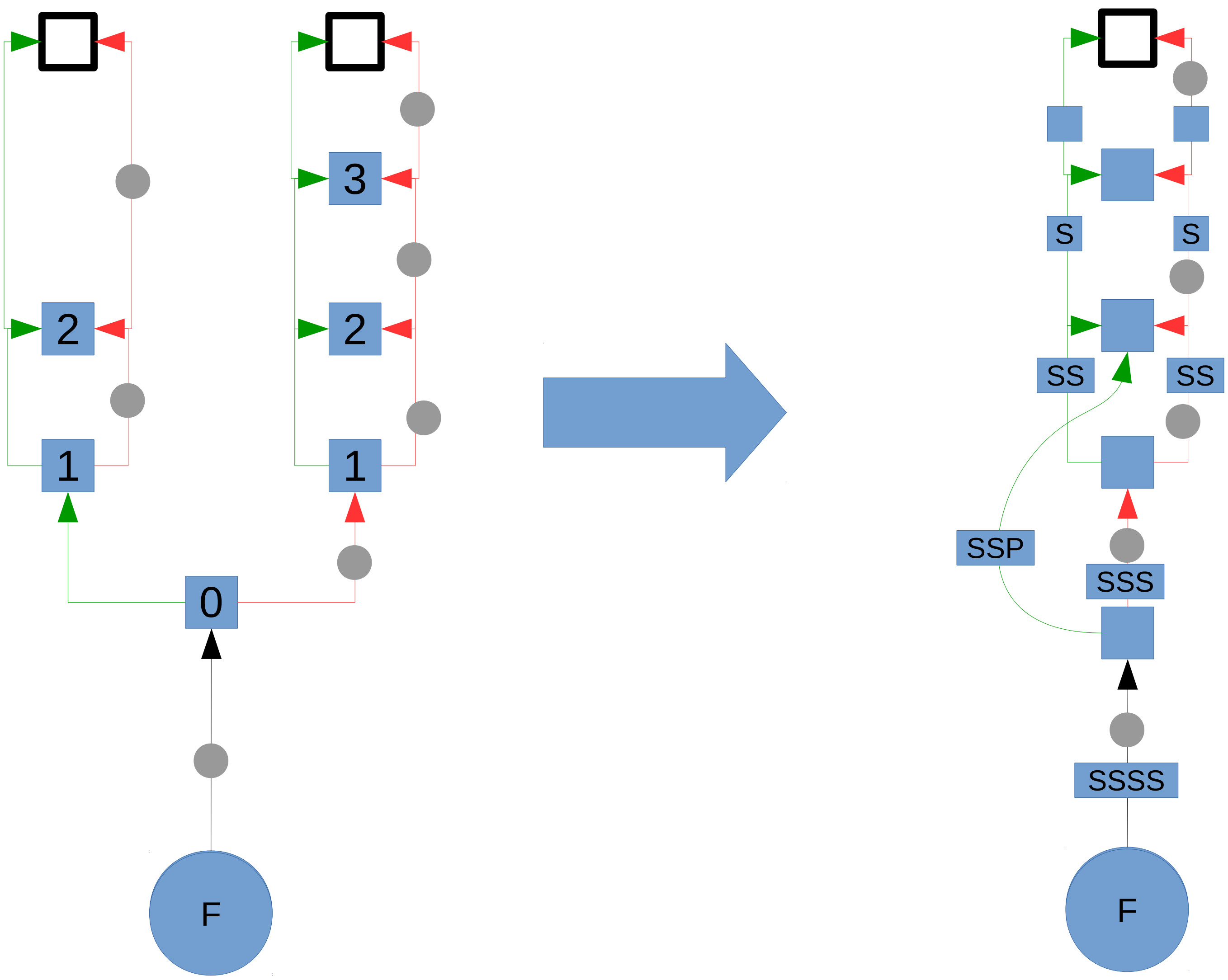




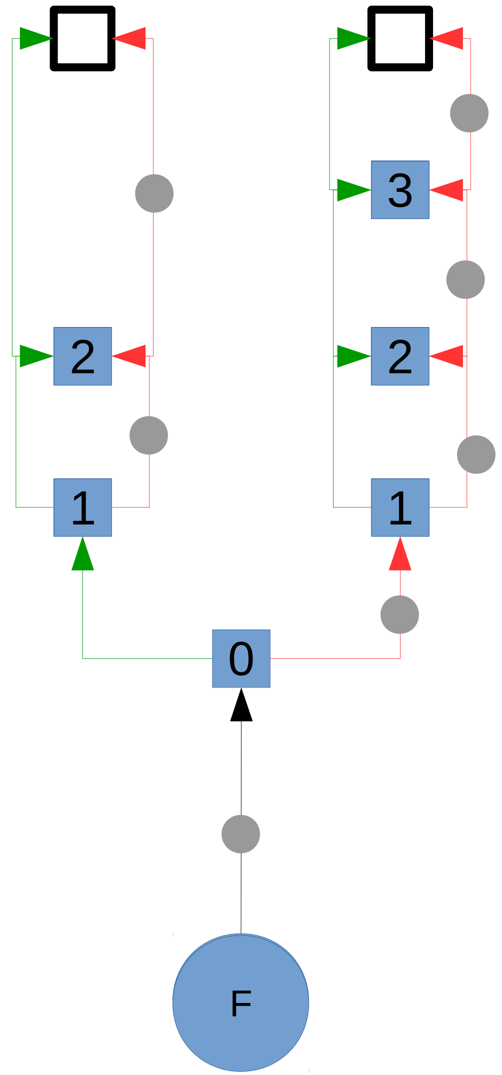
Objectives

- Reduce #node
- Reduce #node’s dependency to variables’ order
- Increase memoization impact

Section 2
Compressing a ROBDD into a GroBdd

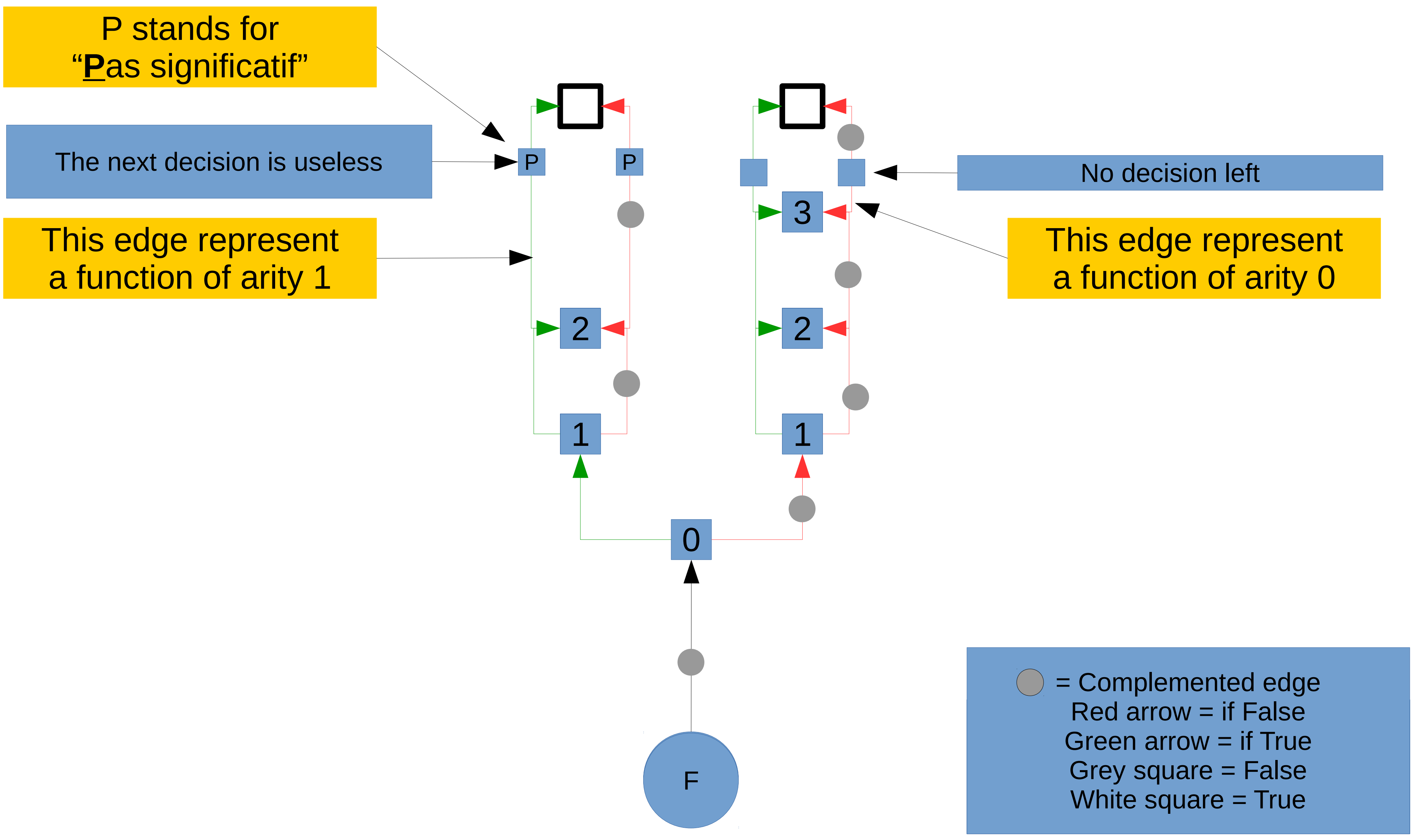


(Model 1) Step 1: for terminal leading edges, we unary represent the number of useless decisions

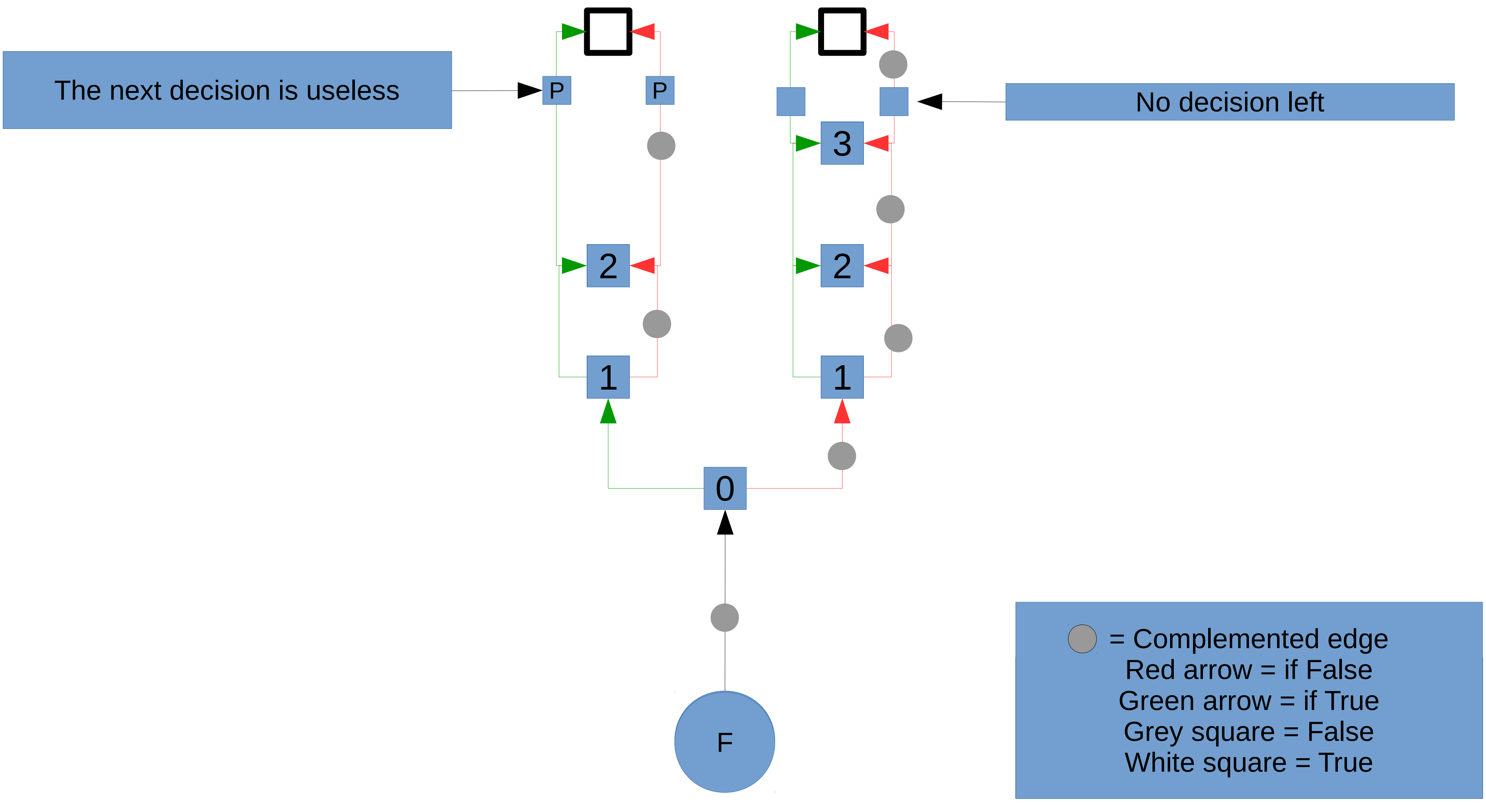


= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

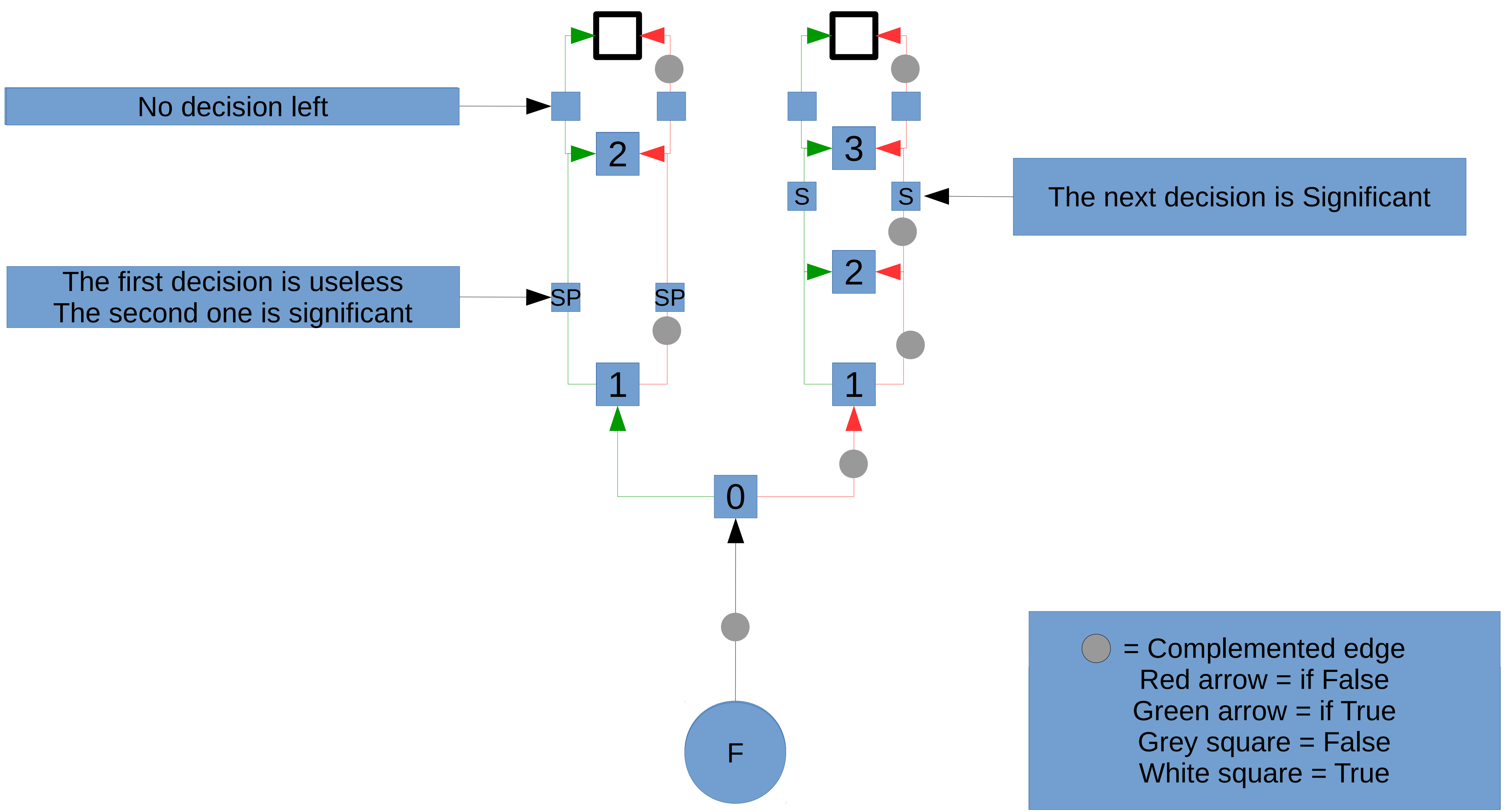
(Model 1) Step 1: for terminal leading edges, we unary represent the number of useless decisions



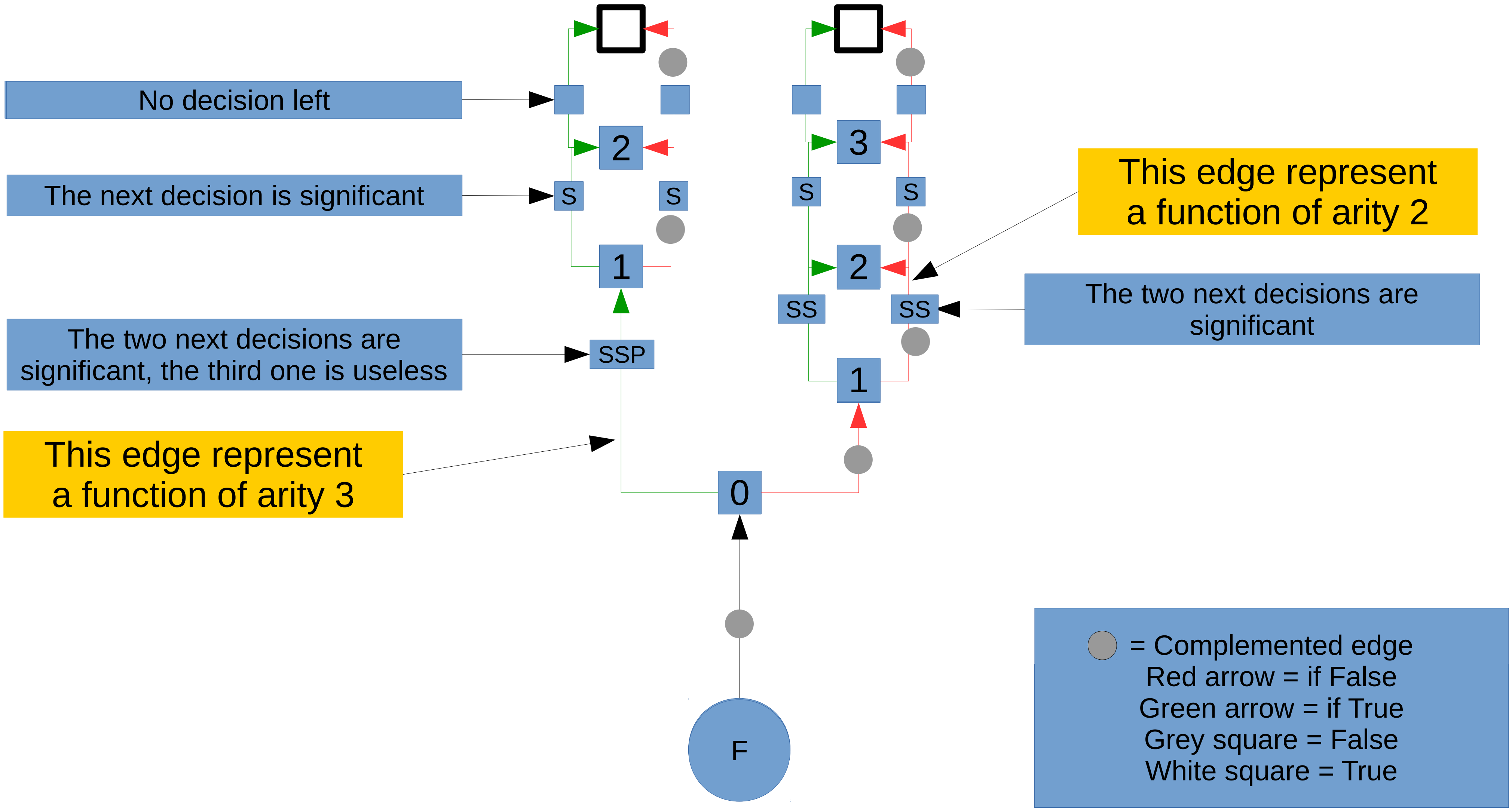
(Model 1) Step 2: we factorize useless variables



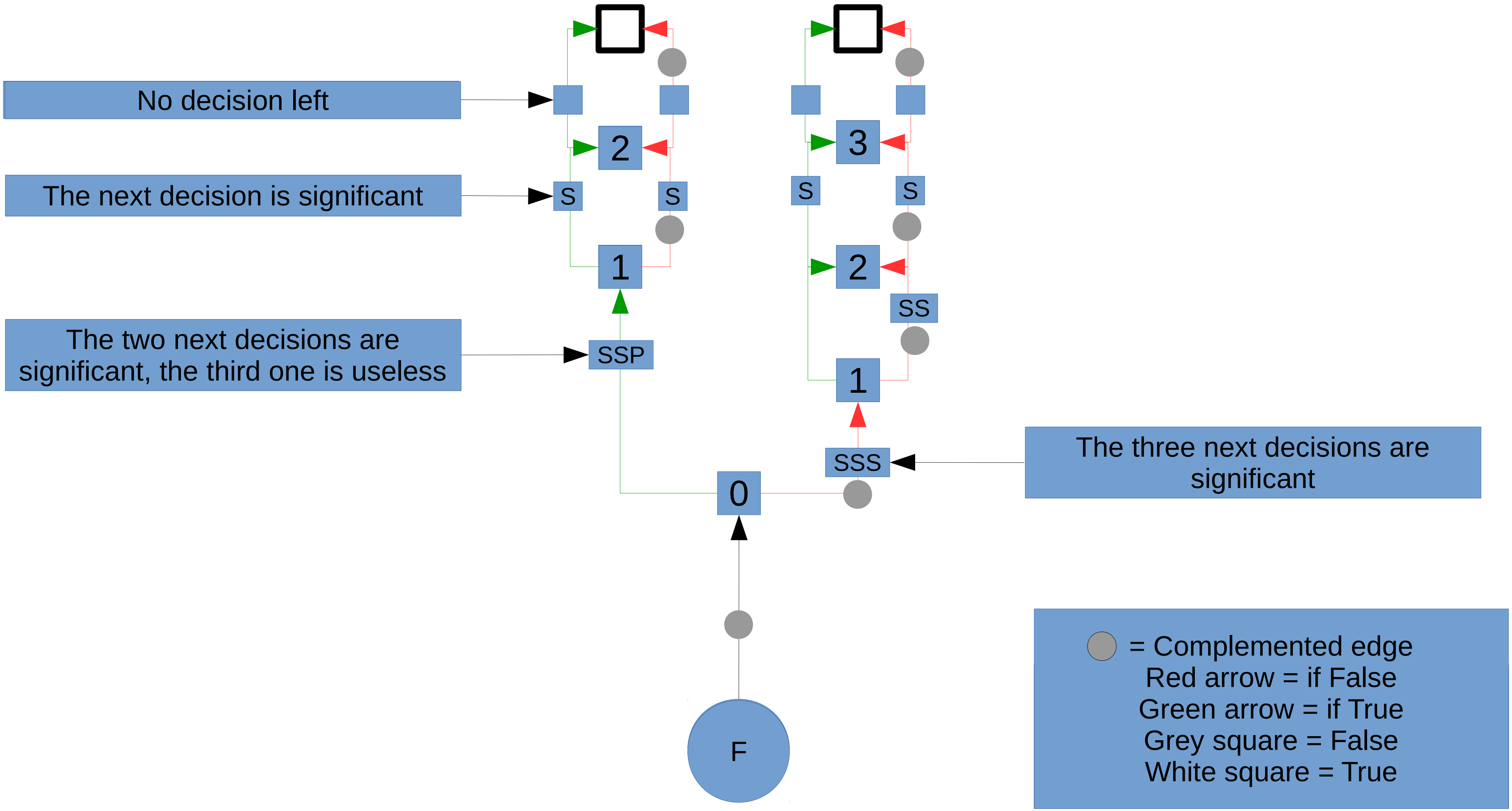
(Model 1) Step 2: we factorize useless variables



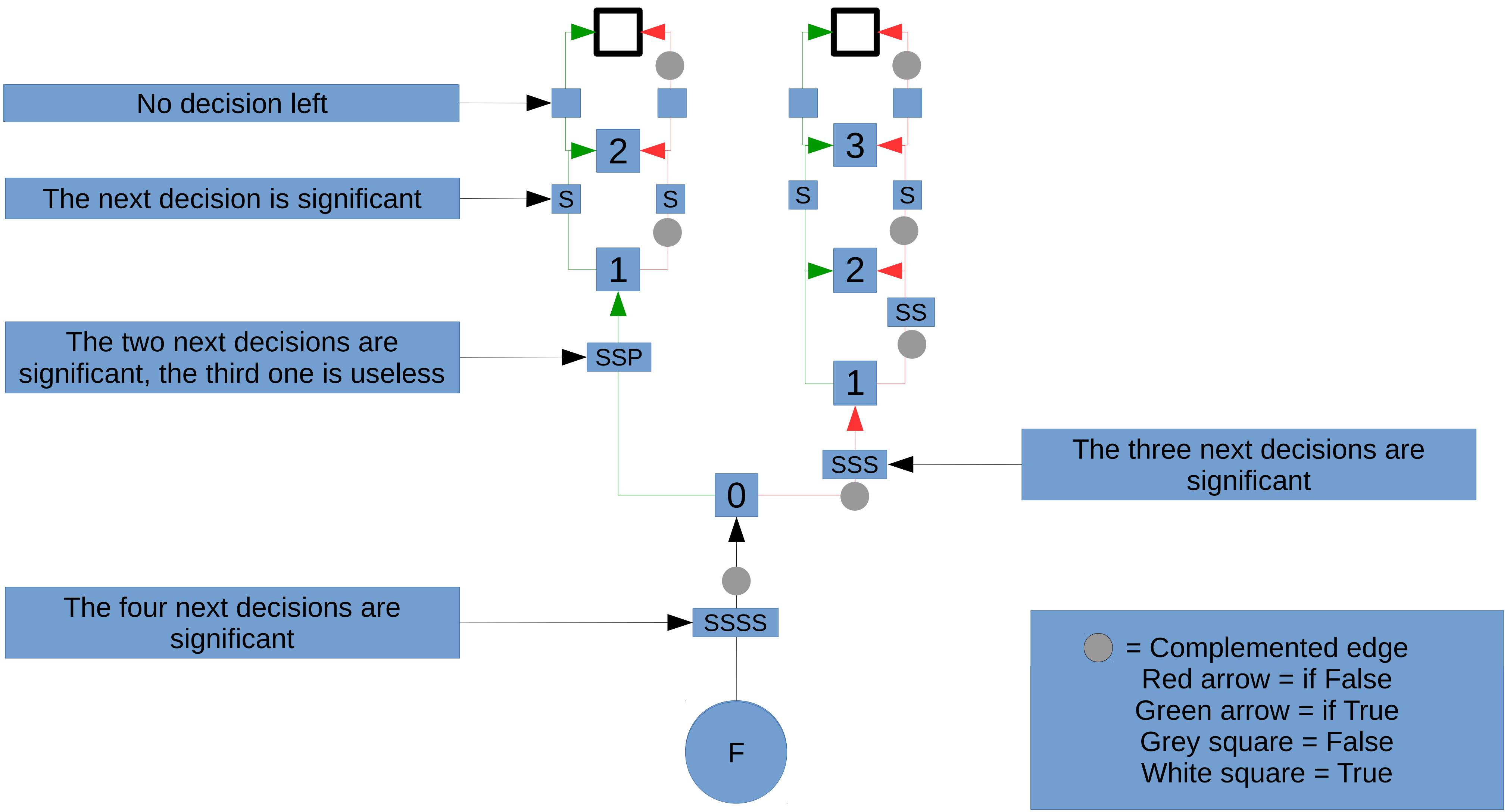
(Model 1) Step 2: we factorize useless variables



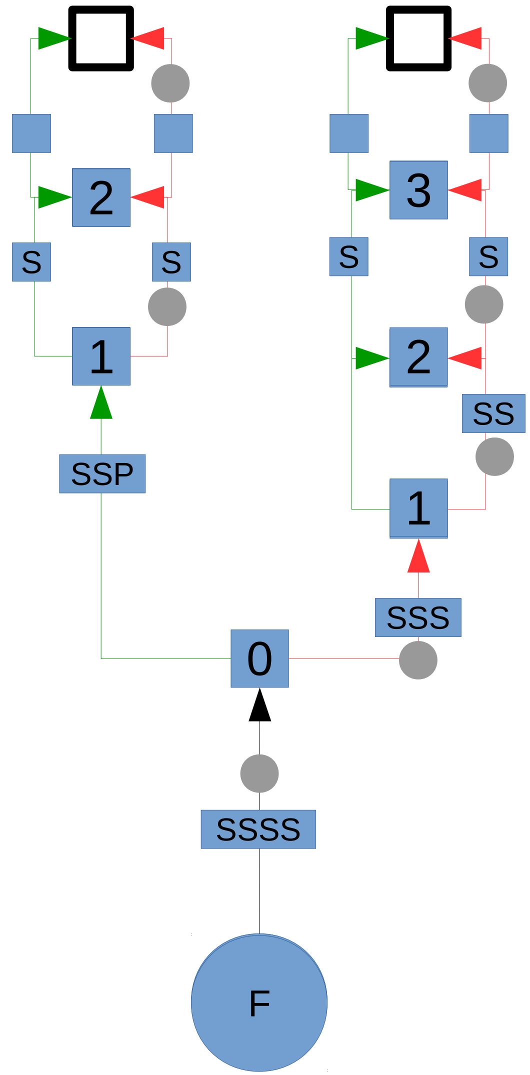
(Model 1) Step 2: we factorize useless variables



(Model 1) Step 2: we factorize useless variables

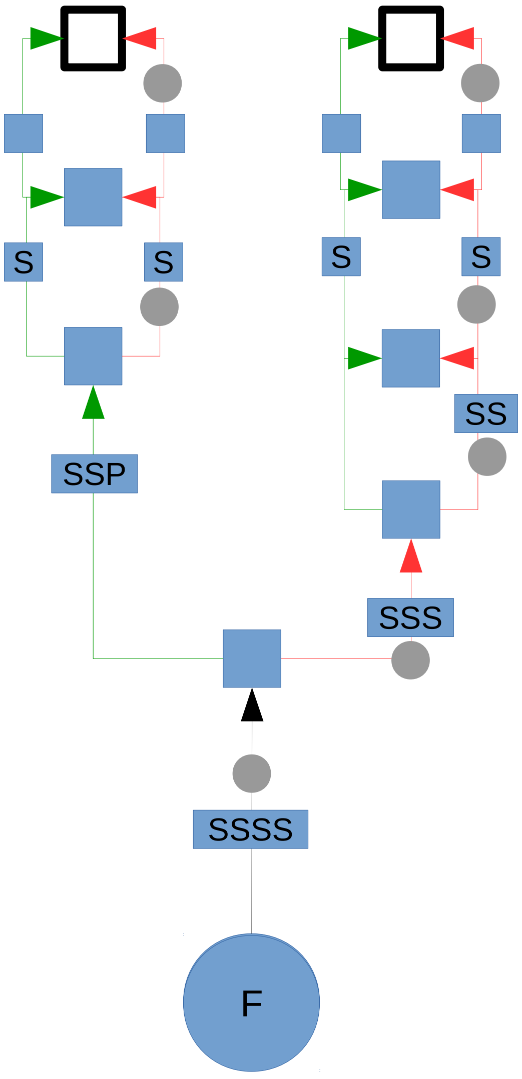


(Model 1) Step 3: we forget every node's depth



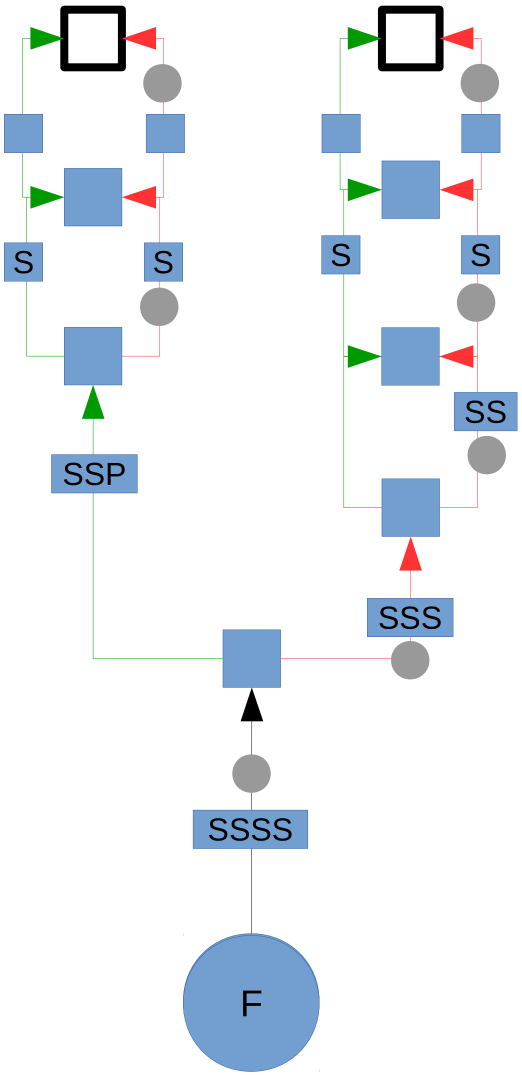
= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

(Model 1) Step 3: we forget every node's depth



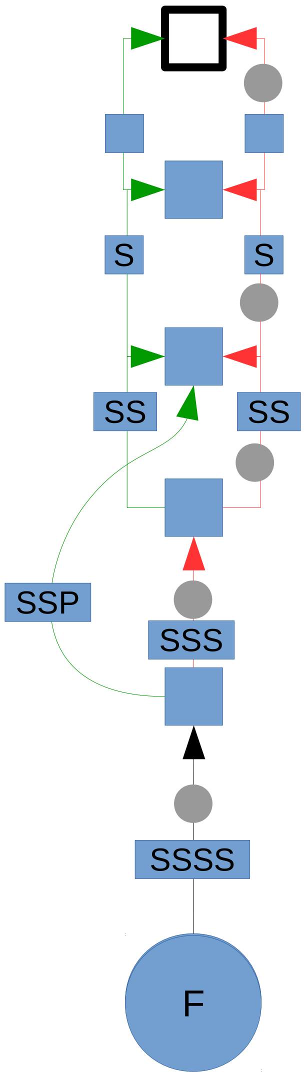
= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

we merge isomorphic sub-graphs



= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

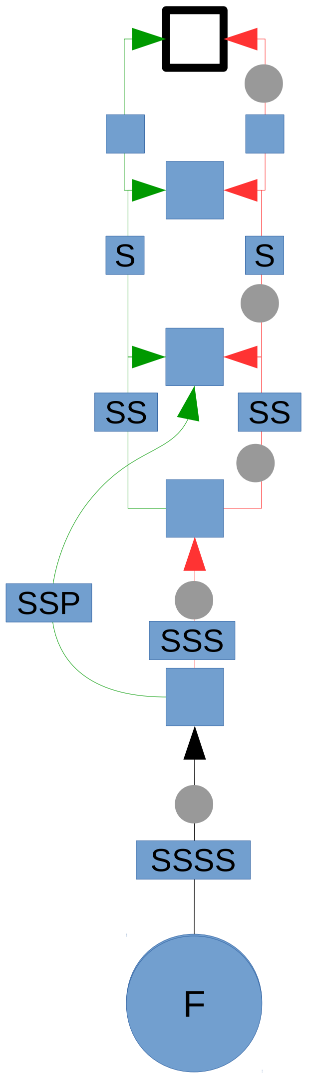
we merge isomorphic sub-graphs



= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

Section 3
Compiling a formula into a GroBdd

$$f(x_0^3) = x_1 \oplus x_2 \oplus (\neg x_0 \wedge x_3)$$



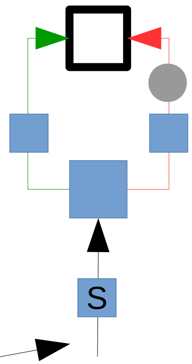
How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1\oplus x_2\oplus\left(\neg x_0\wedge x_3\right)$$

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f(x_0^3) = x_1 \oplus x_2 \oplus (\neg x_0 \wedge x_3)$$

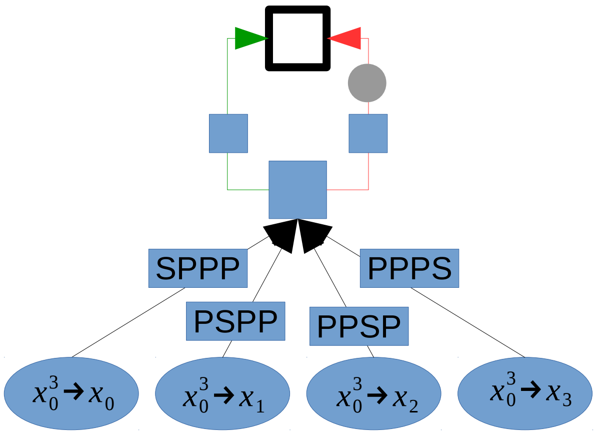


Step 1: we build
the identity function

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right) = x_1 \oplus x_2 \oplus \left(\neg x_0 \wedge x_3\right)$$

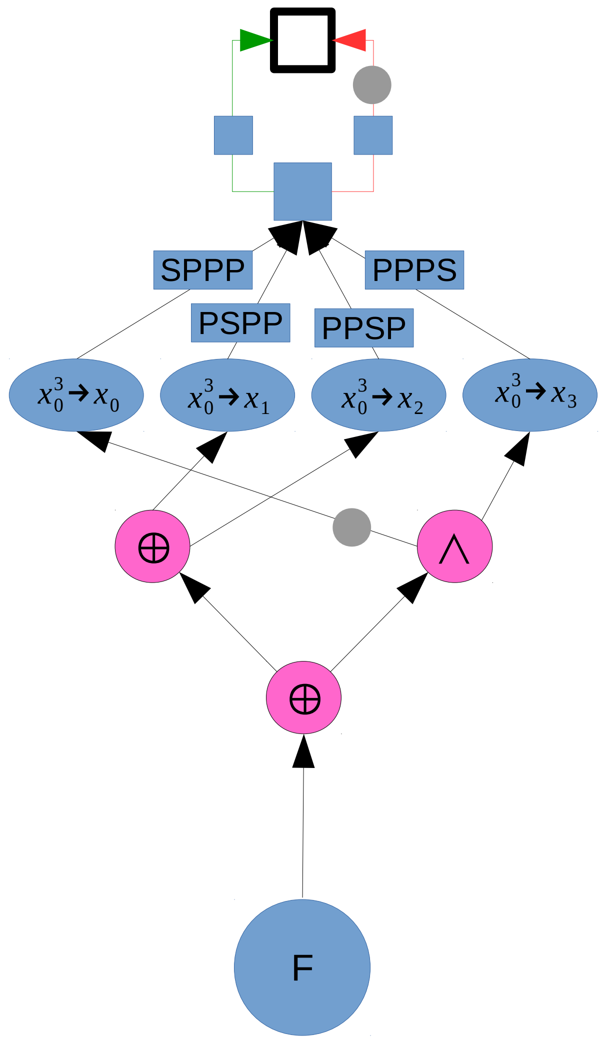


Step 2: we build one projection per variable

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1\oplus x_2\oplus\left(\neg x_0\wedge x_3\right)$$

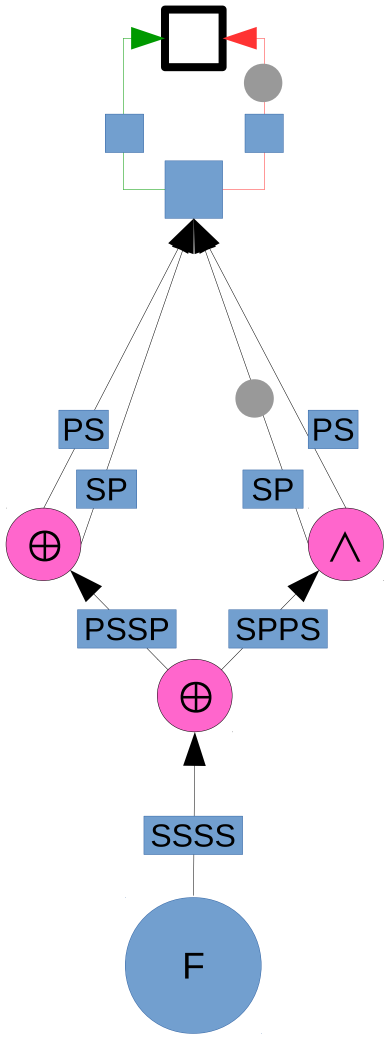


Step 3: we build the formula

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

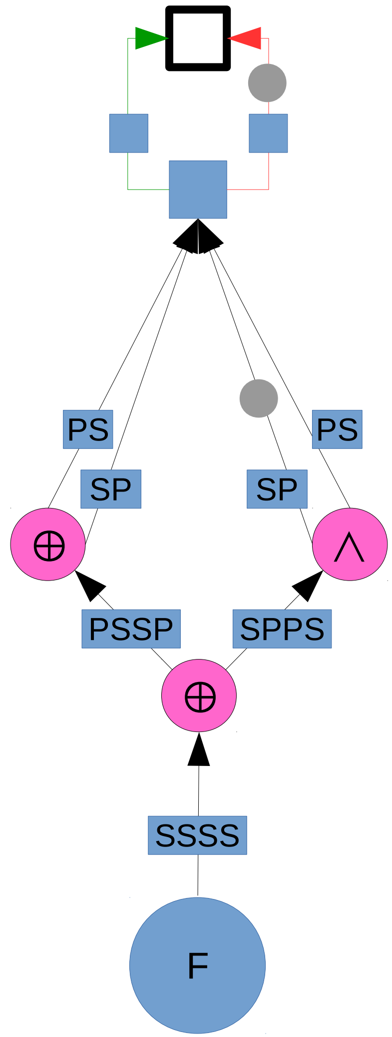
$$f\left(x_0^3\right)=x_1 \oplus x_2 \oplus \left(\neg x_0 \wedge x_3\right)$$



Step 4: we factorize useless variables

How to compile a formula into a GroBdd

$$f(x_0^3) = x_1 \oplus x_2 \oplus (\neg x_0 \wedge x_3)$$

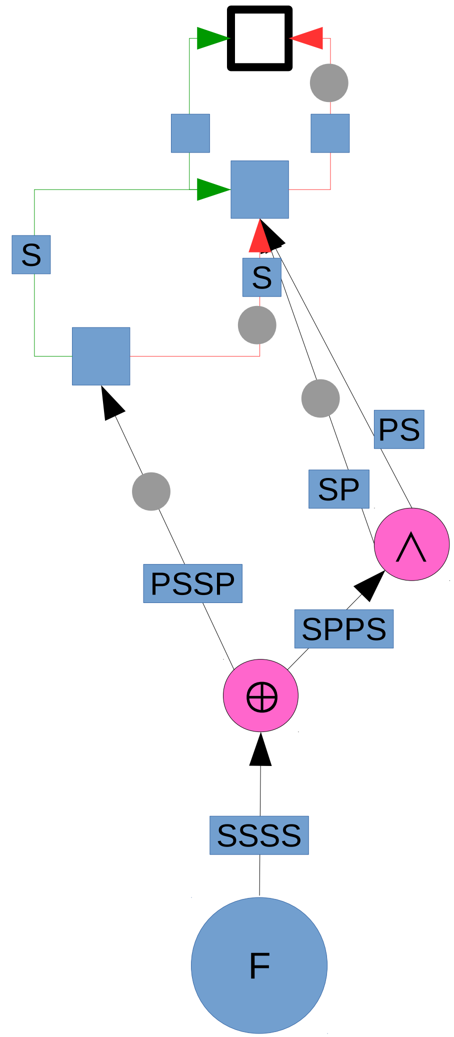


Step 5: we compute operator nodes

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1 \oplus x_2 \oplus \left(\neg x_0 \wedge x_3\right)$$

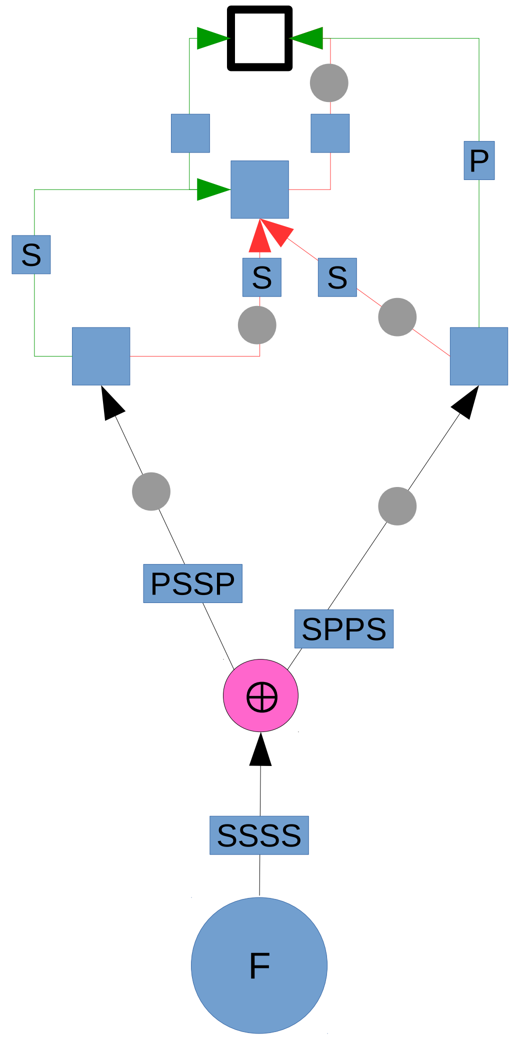


Step 5: we compute operator nodes

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1 \oplus x_2 \oplus \left(\neg x_0 \wedge x_3\right)$$

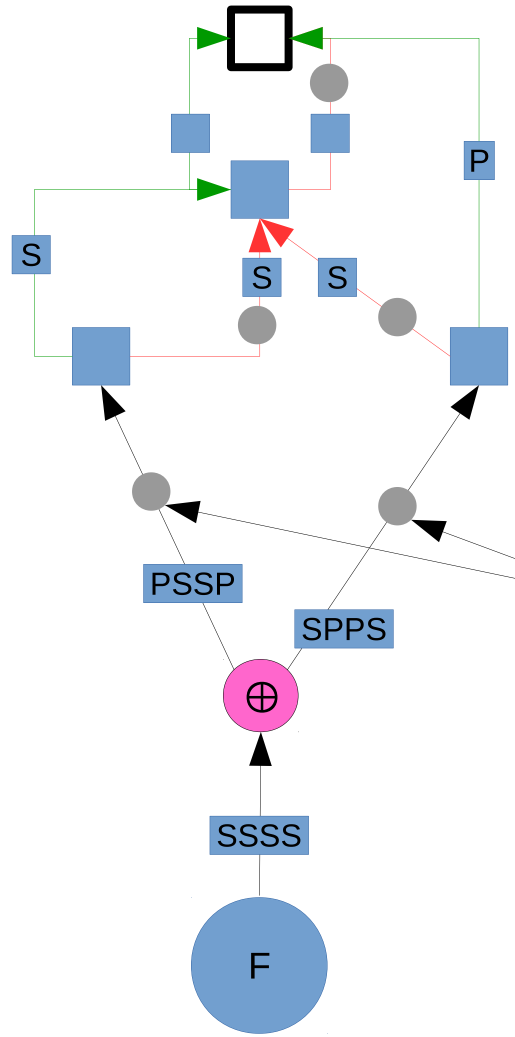


Step 5: we compute operator nodes

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1\oplus x_2\oplus\left(\neg x_0\wedge x_3\right)$$



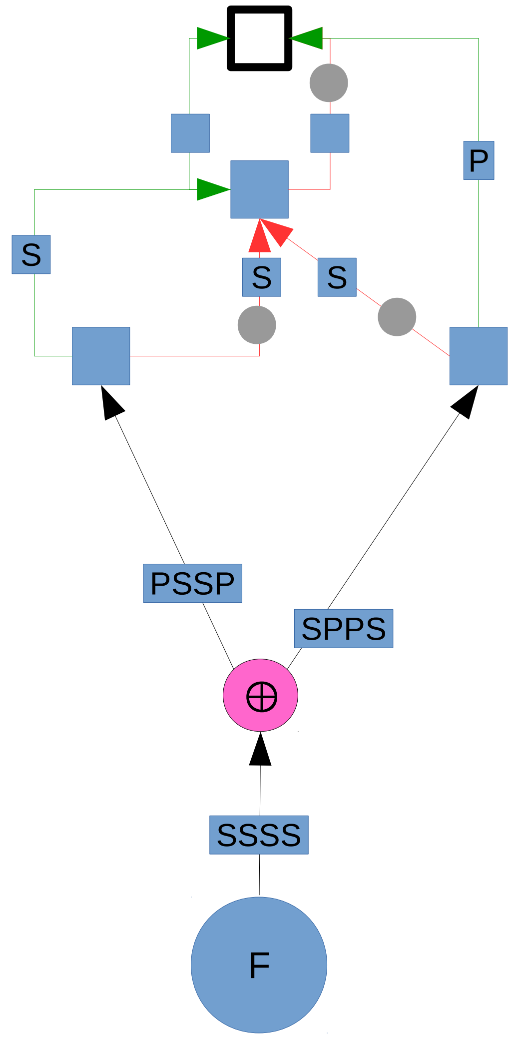
Step 5: we compute operator nodes

$$\neg f\oplus\neg g=f\oplus g$$

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1 \oplus x_2 \oplus\left(\neg x_0 \wedge x_3\right)$$

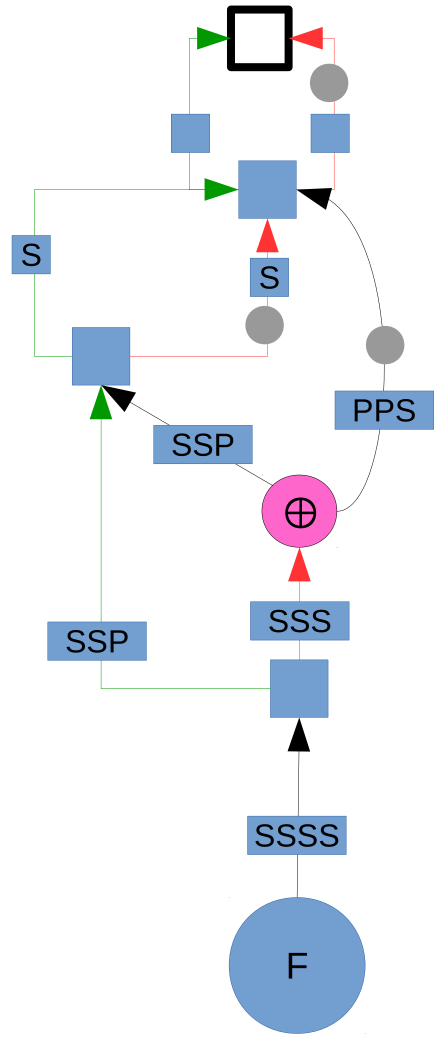


Step 5: we compute operator nodes

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

$$f\left(x_0^3\right)=x_1\oplus x_2\oplus\left(\neg x_0\wedge x_3\right)$$

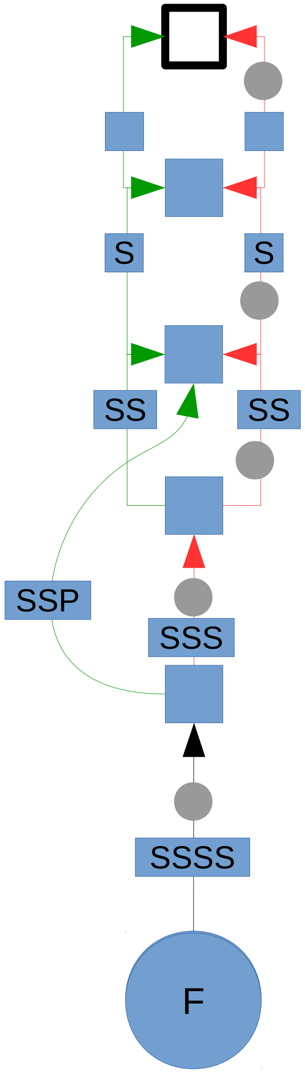


Step 5: we compute operator nodes

= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

How to compile a formula into a GroBdd

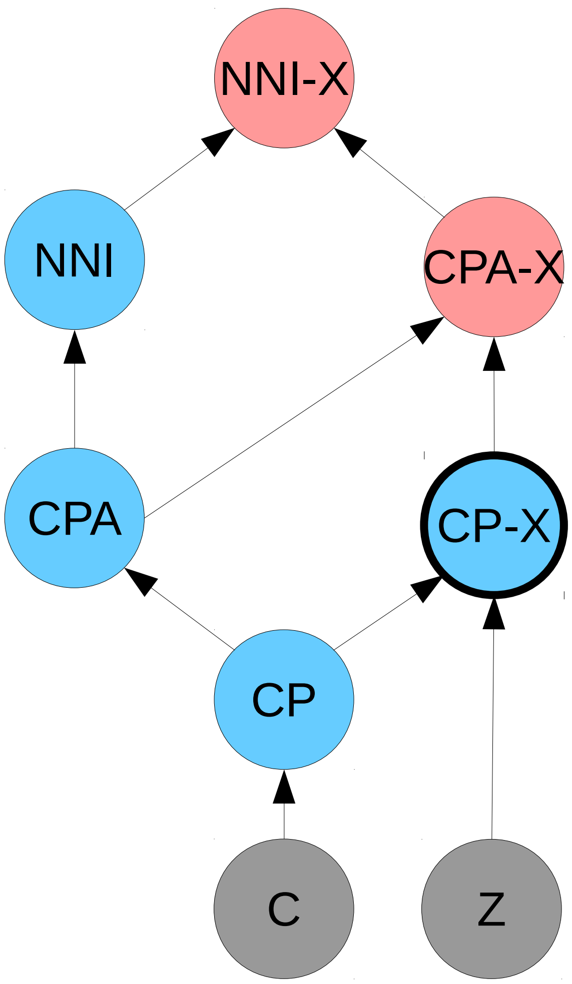
$$f(x_0^3) = x_1 \oplus x_2 \oplus (\neg x_0 \wedge x_3)$$

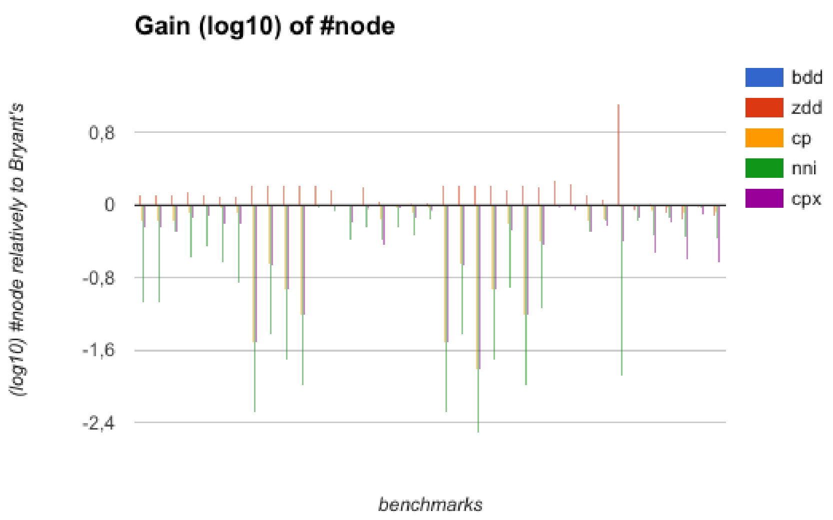


Step 5: we compute operator nodes

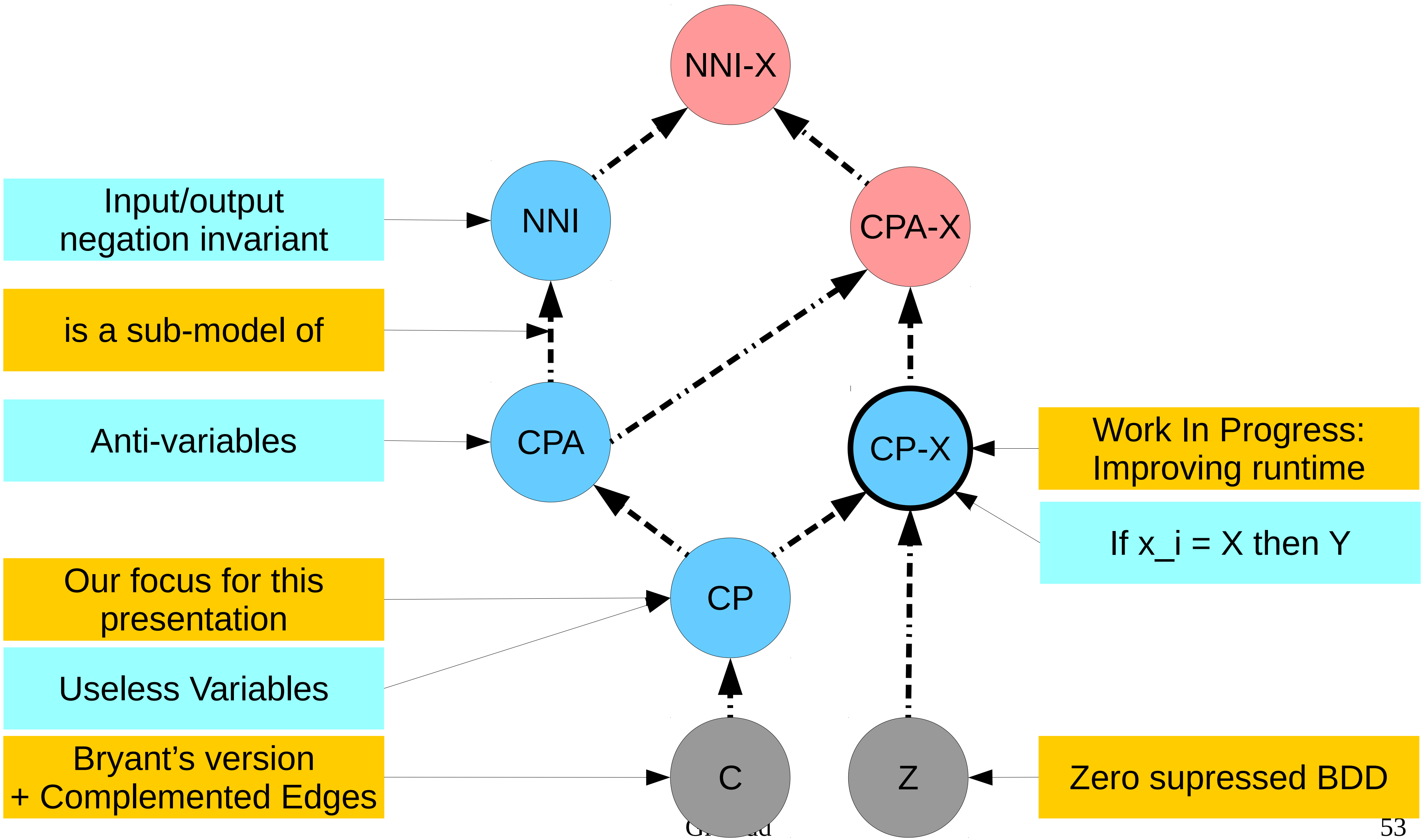
= Complemented edge
Red arrow = if False
Green arrow = if True
Grey square = False
White square = True

Section 4
Generalization & Results

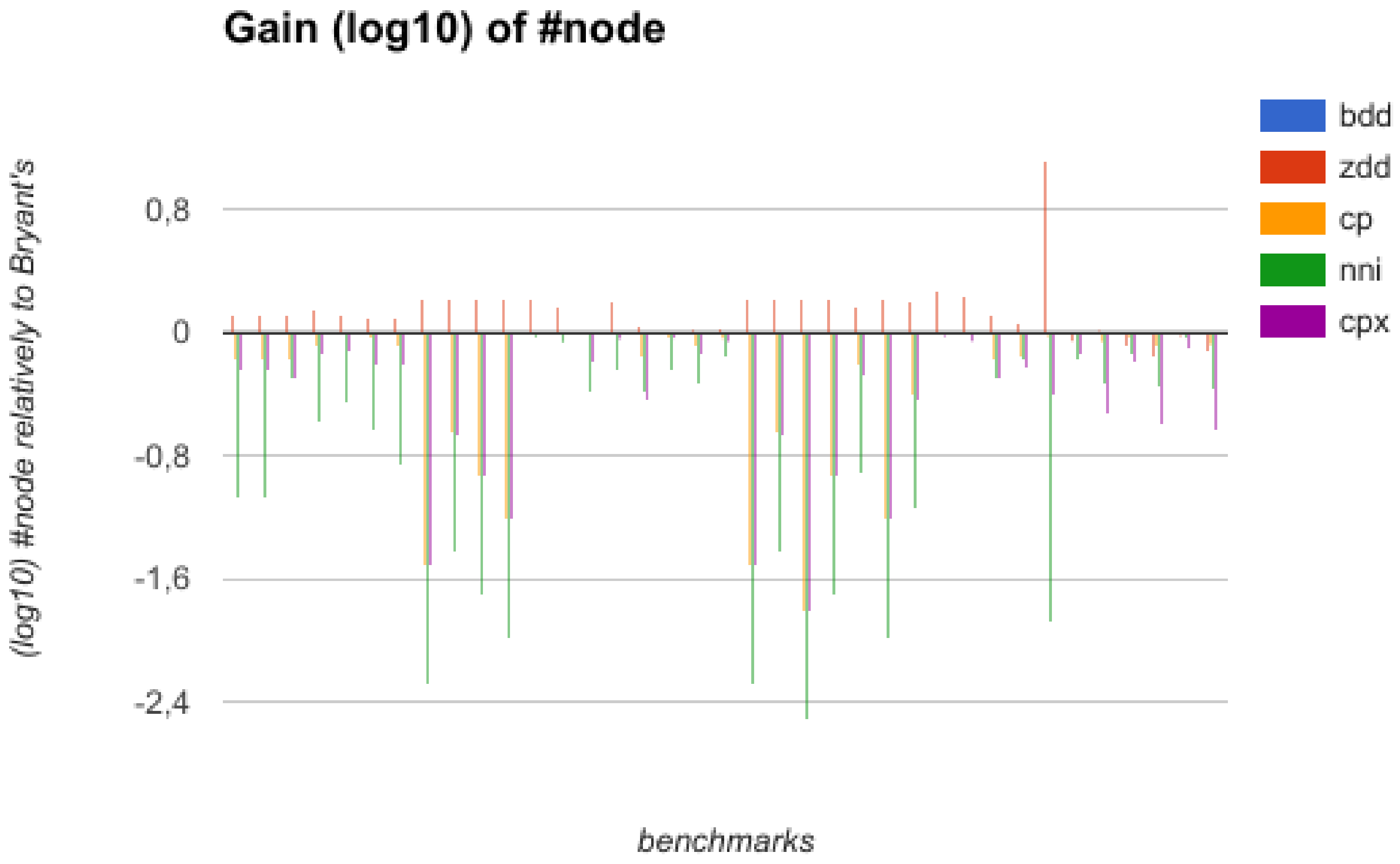




Some Generalised Reductions

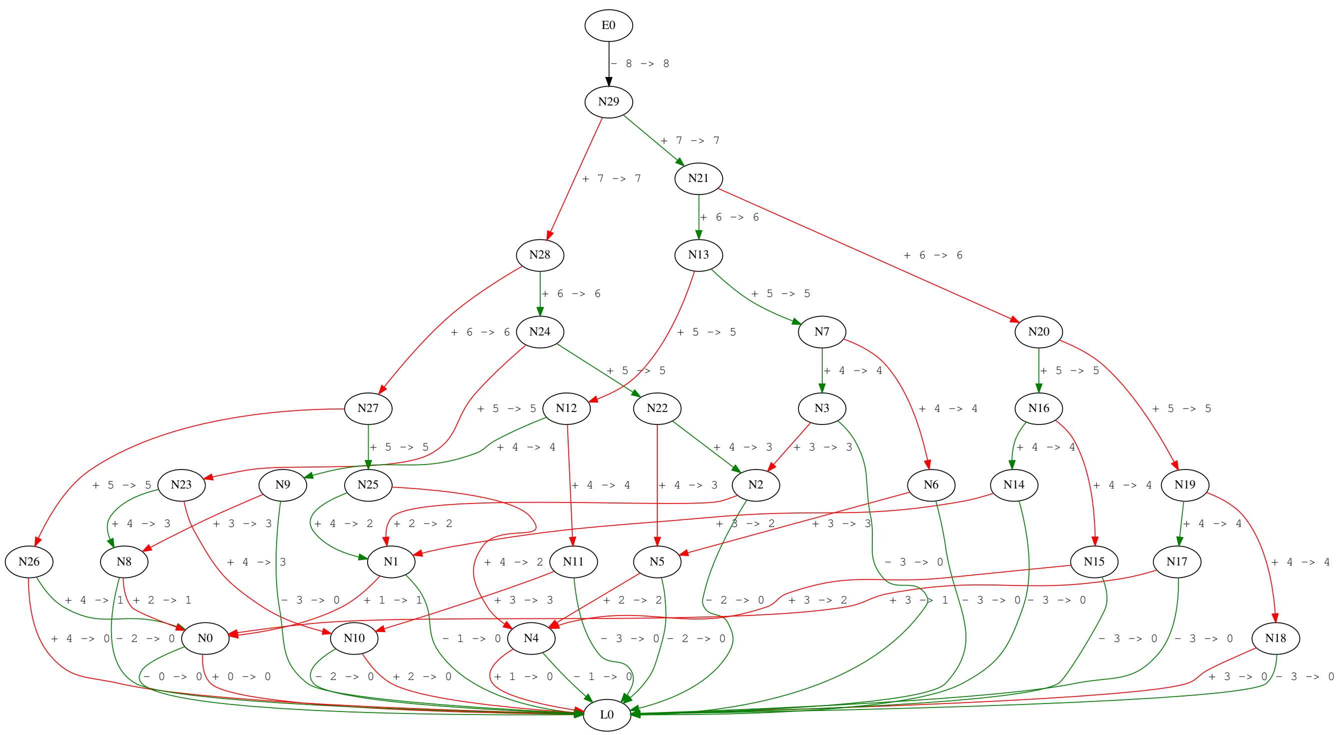


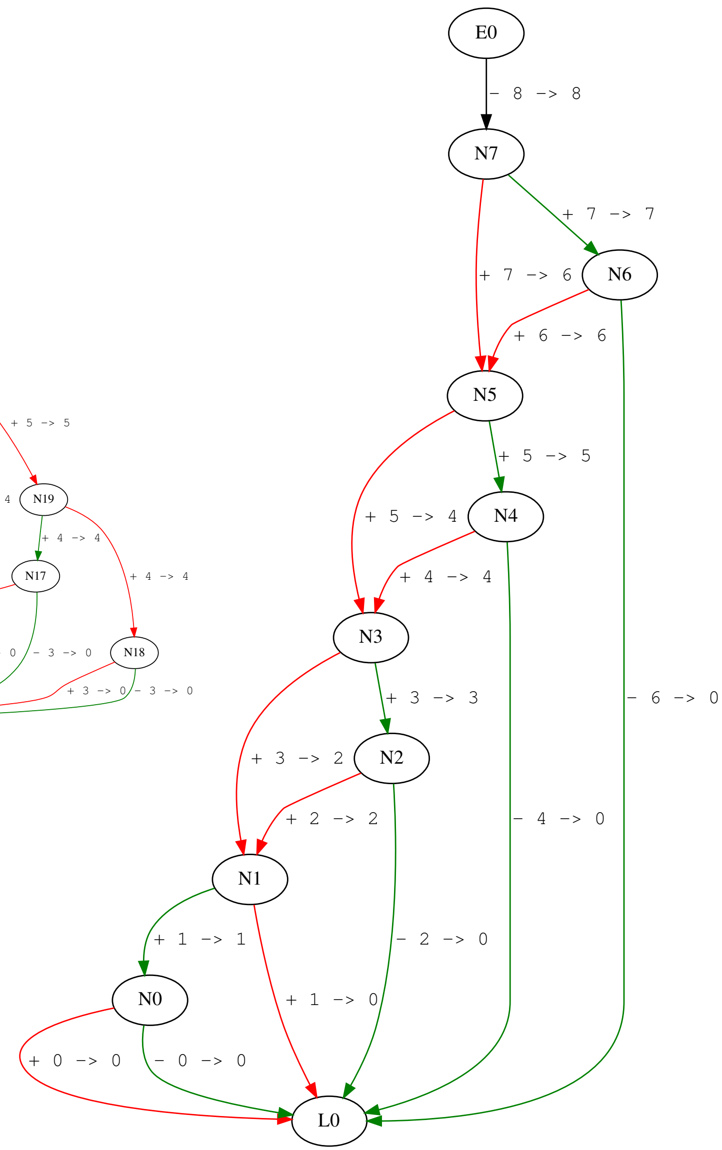
Results



NB: I'm still working on how to show results

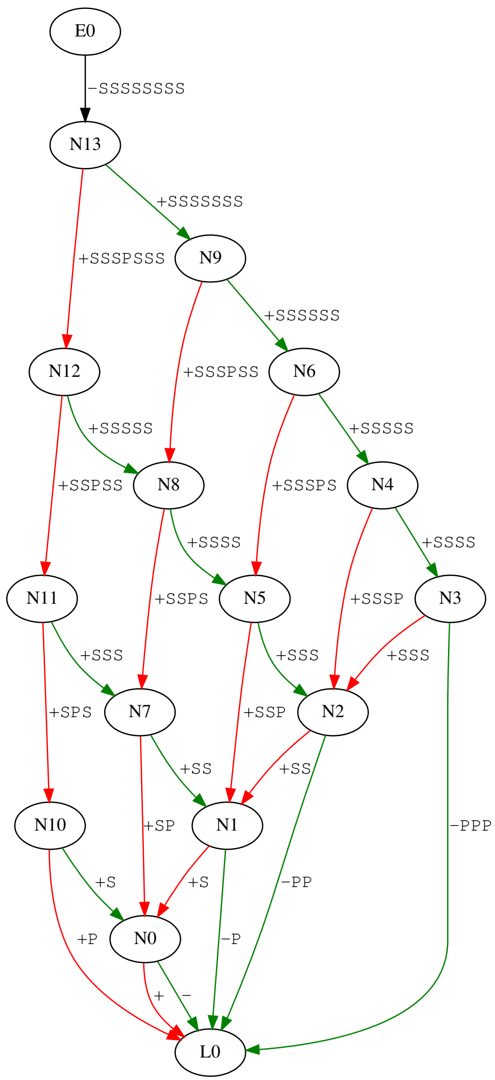
#node is order dependent

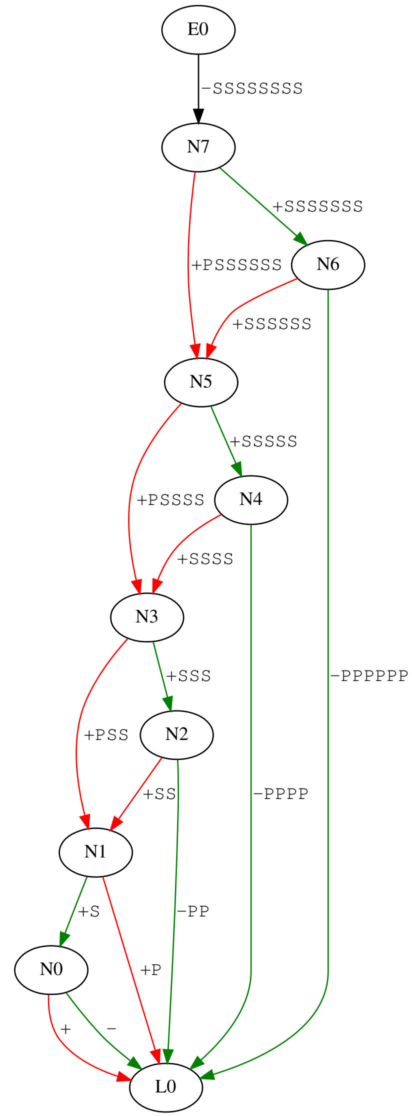




Bryant’s version

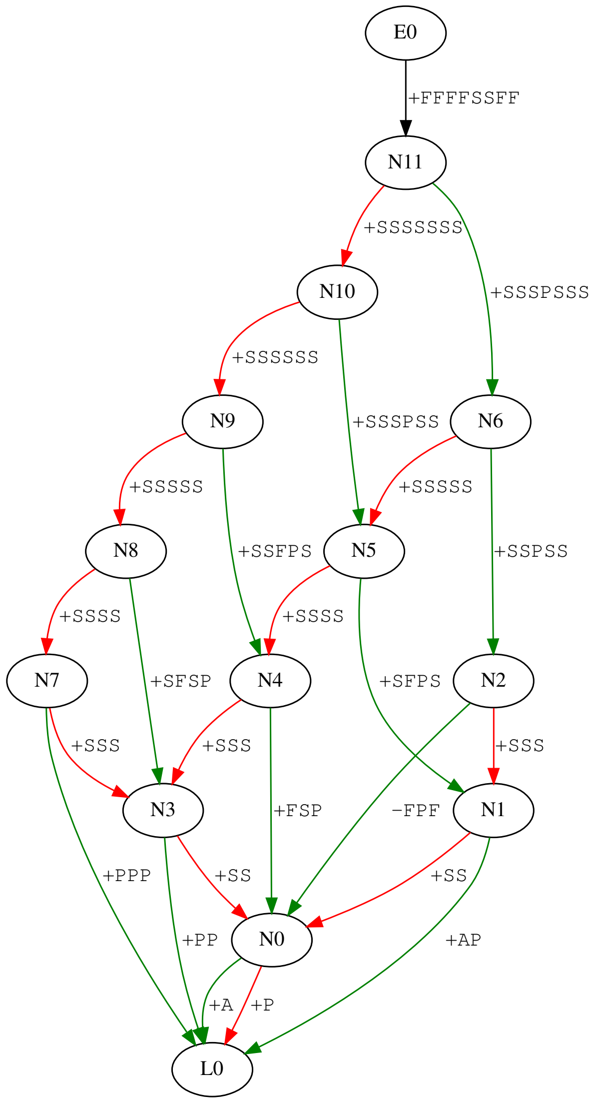
#node is order dependent

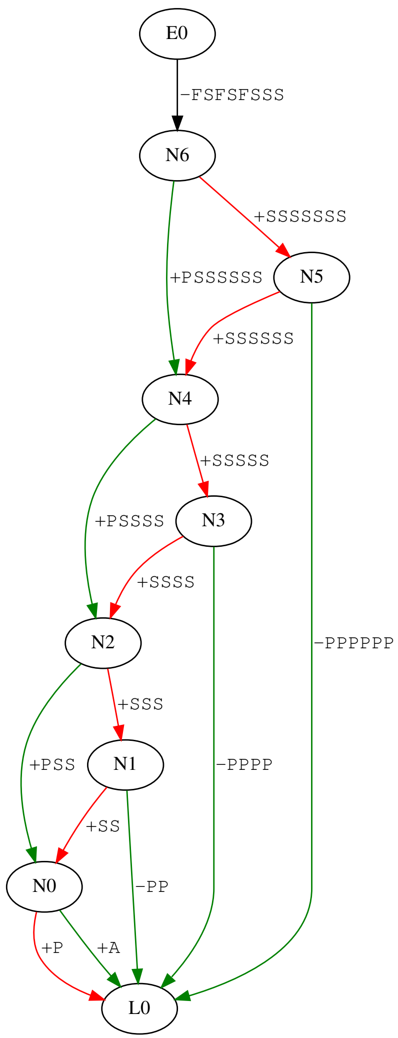




CP version

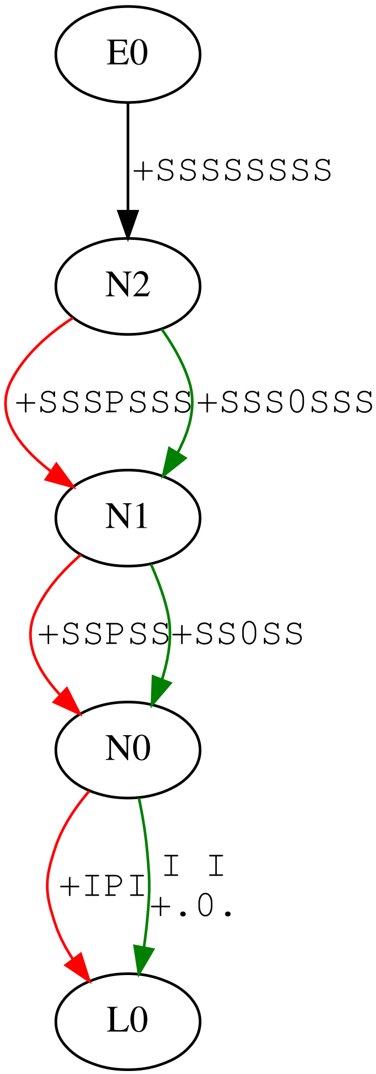
#node is order dependent

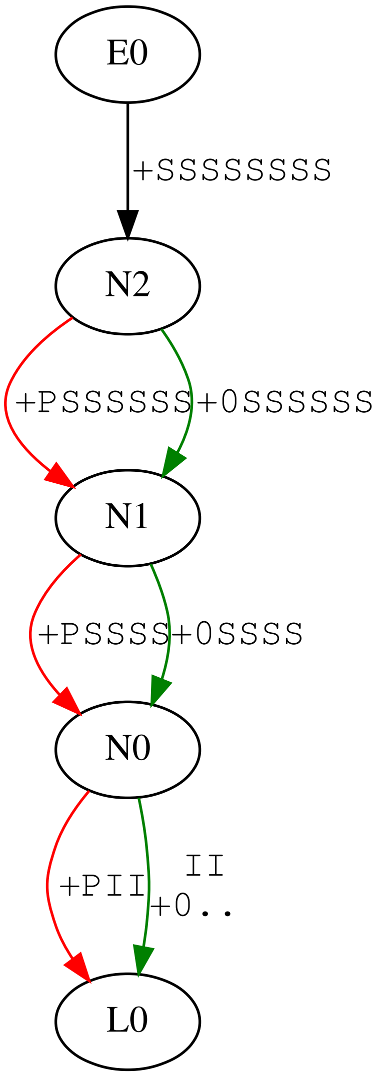




NNI version

#node is order dependent





CP-X version

Conclusion

- Software implemented in OCaml:
  - https://github.com/JoanThibault/DAGaml/tree/grobdd-dev
  - ~ 10 000 lines of OCaml
- Fewer nodes
  - CP : -0.35 d (-55%)
  - NNI : -0.51 d (-69%)
  - CPX : -0.13 d (-26%)
- Future Work
  - Quantify the dependency between variables' order and #node
  - Solve & Implement CPA-X and NNI-X versions
- TO DO
  - Parallelism & hardware acceleration
  - Quantification Operators
  - Variable Reordering