Some Generalised Reductions of Ordered Binary Decision Diagramm (GroBdd)

Joan Thibault

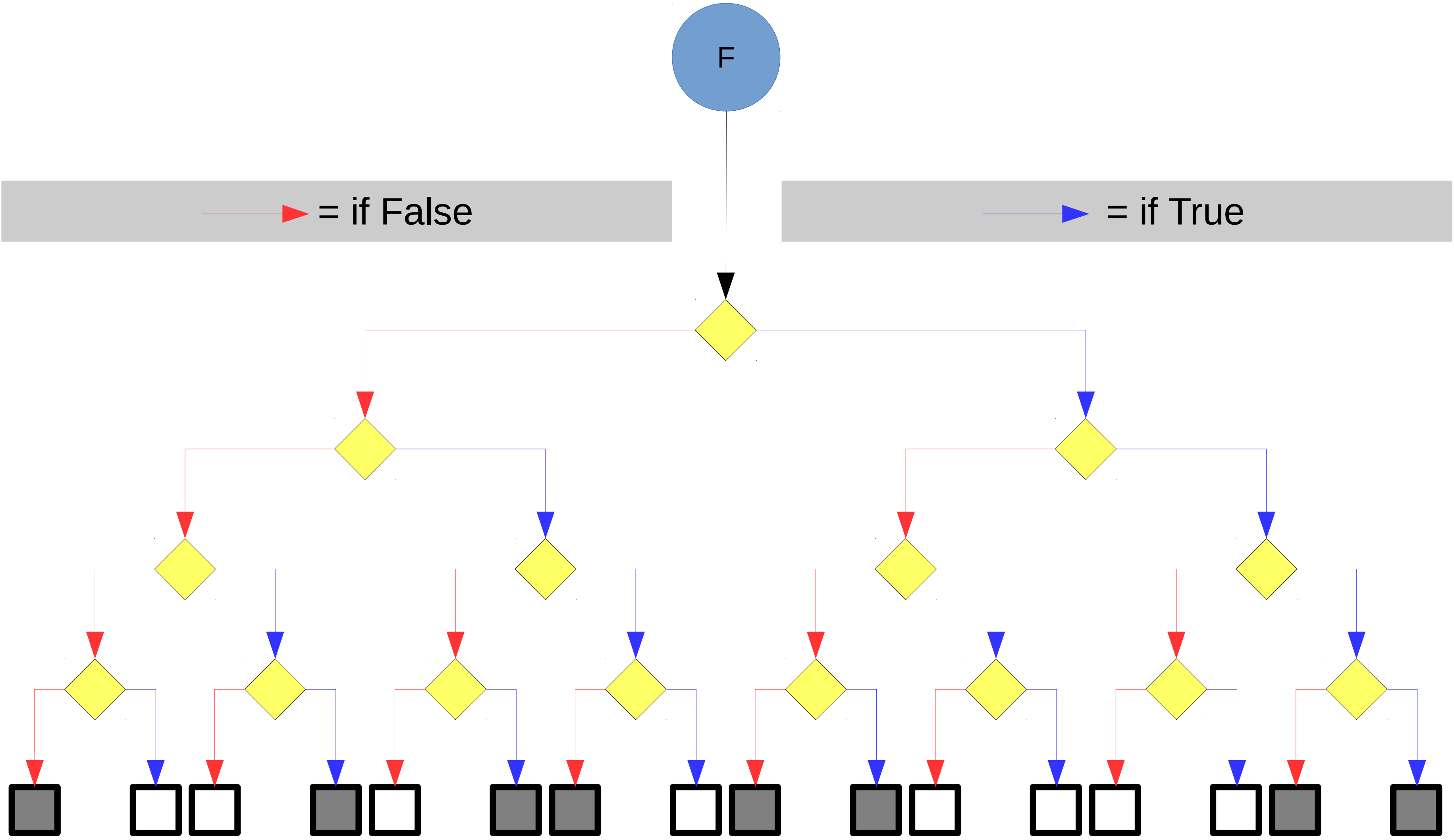
Boolean Functions

- 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 ?



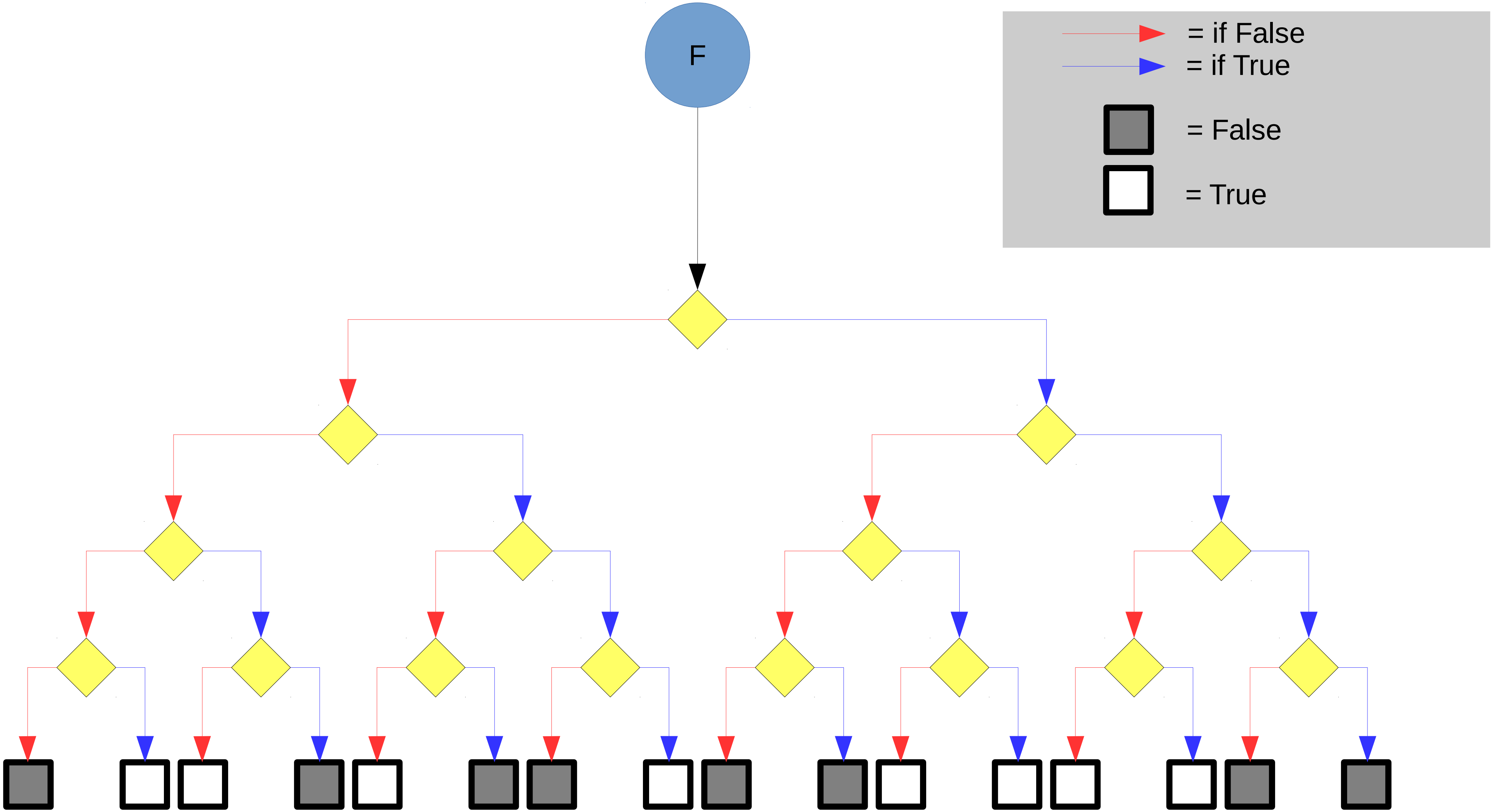
→ = if False

→ = if True

▨ = False

□ = True

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



= if False

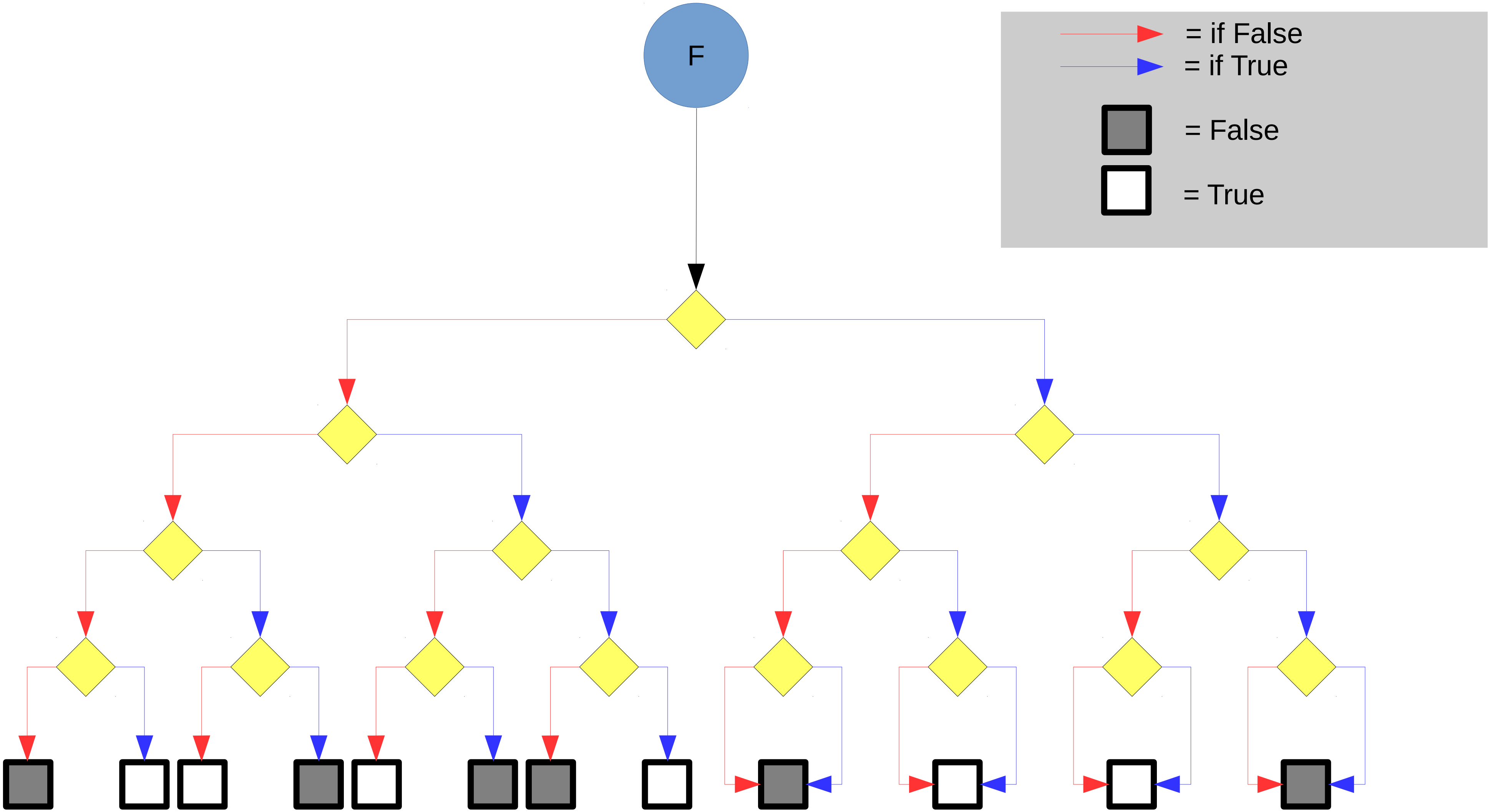
= if True

= False

= True

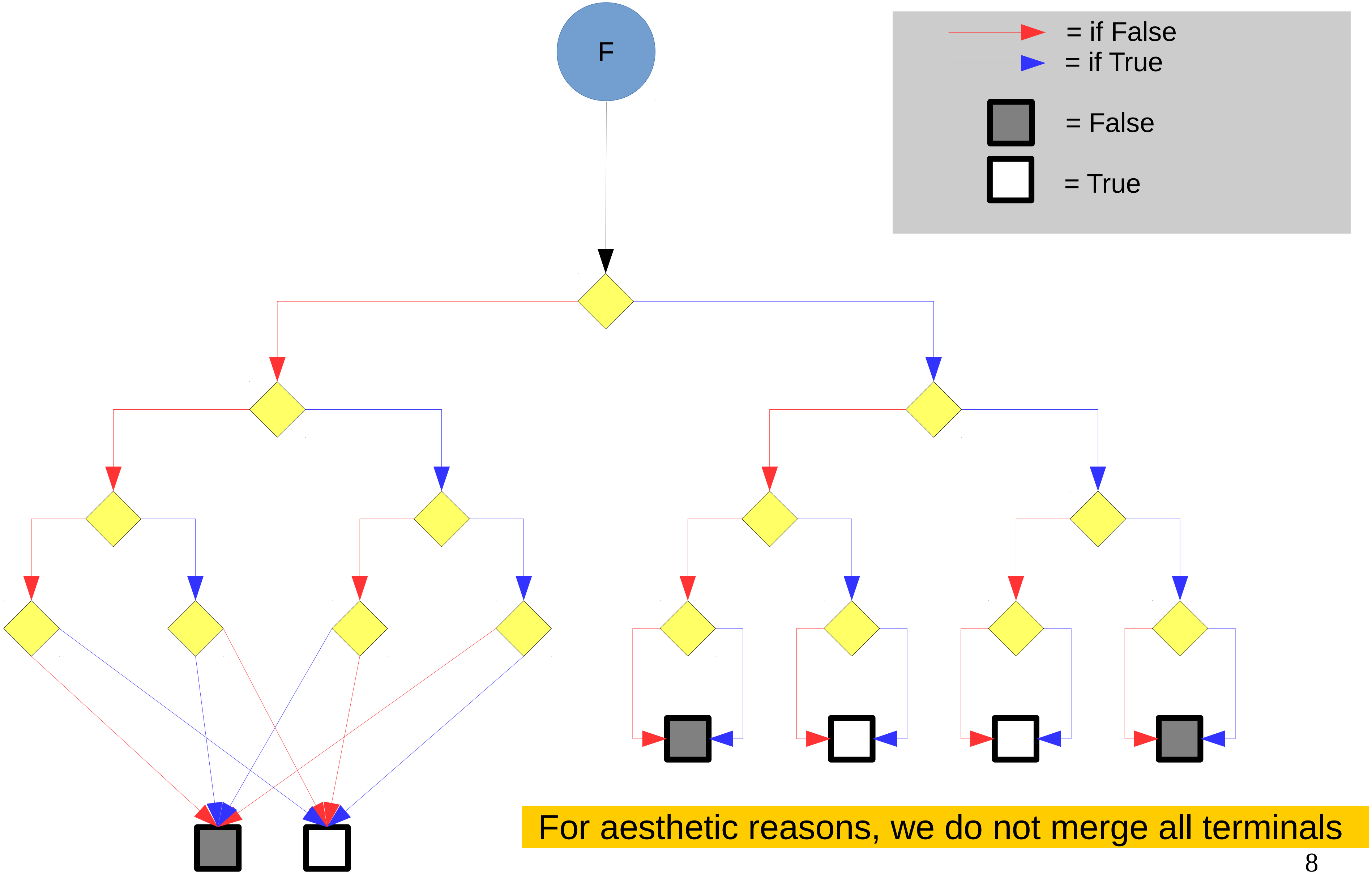
For aesthetic reasons, we do not merge all terminals

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



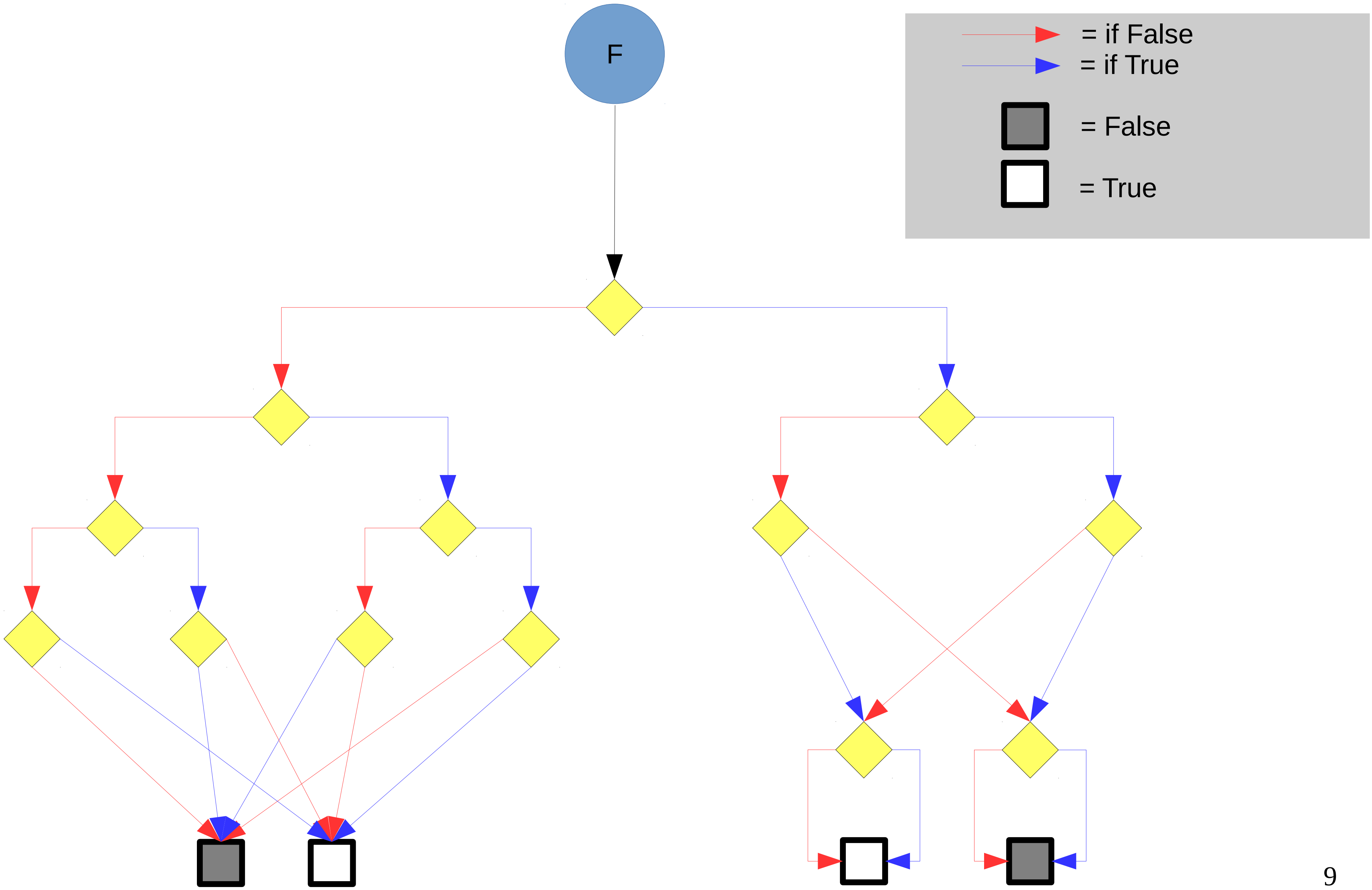
For aesthetic reasons, we do not merge all terminals

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



For aesthetic reasons, we do not merge all terminals

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



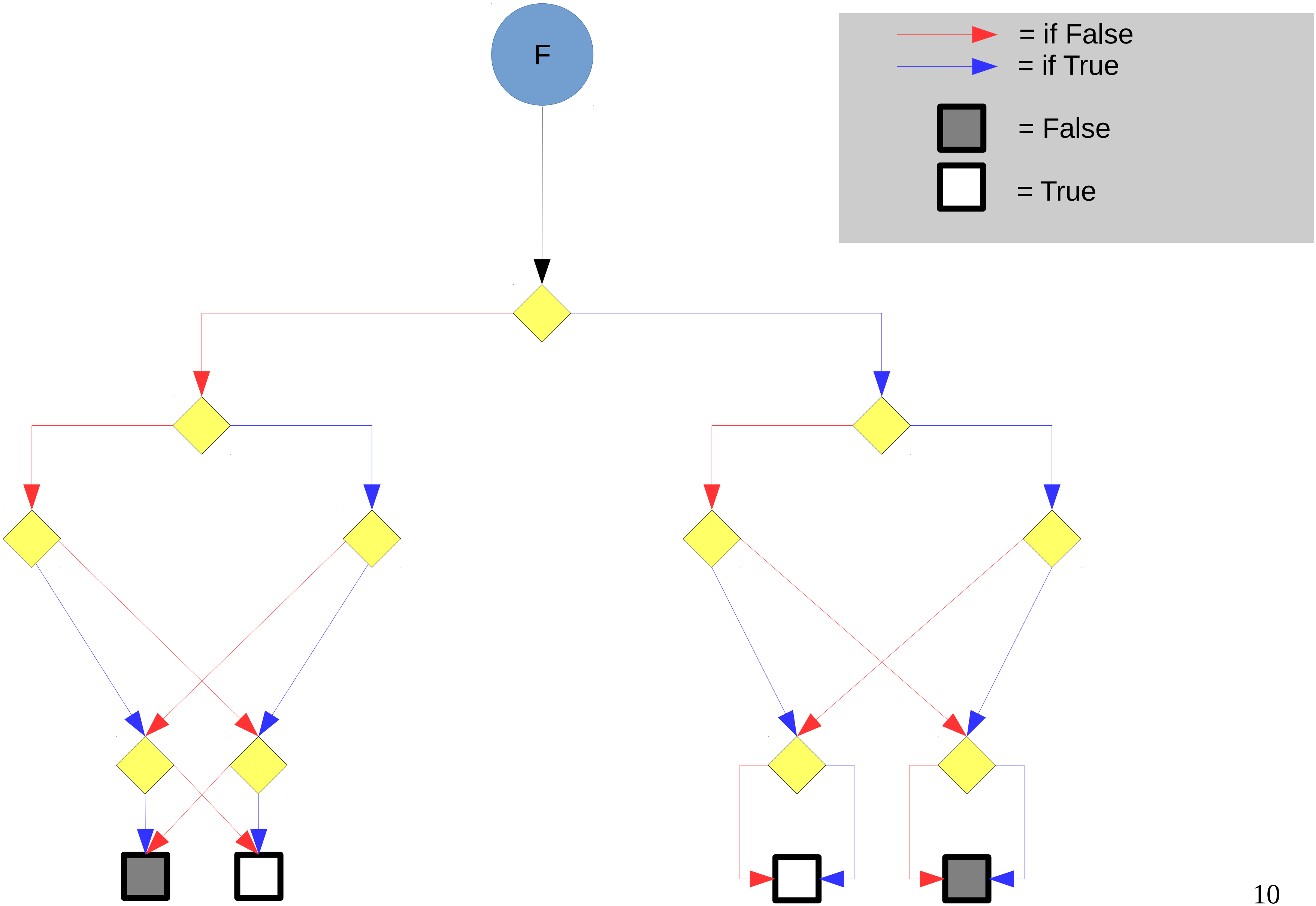
→ = if False

→ = if True

■ = False

□ = True

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



F

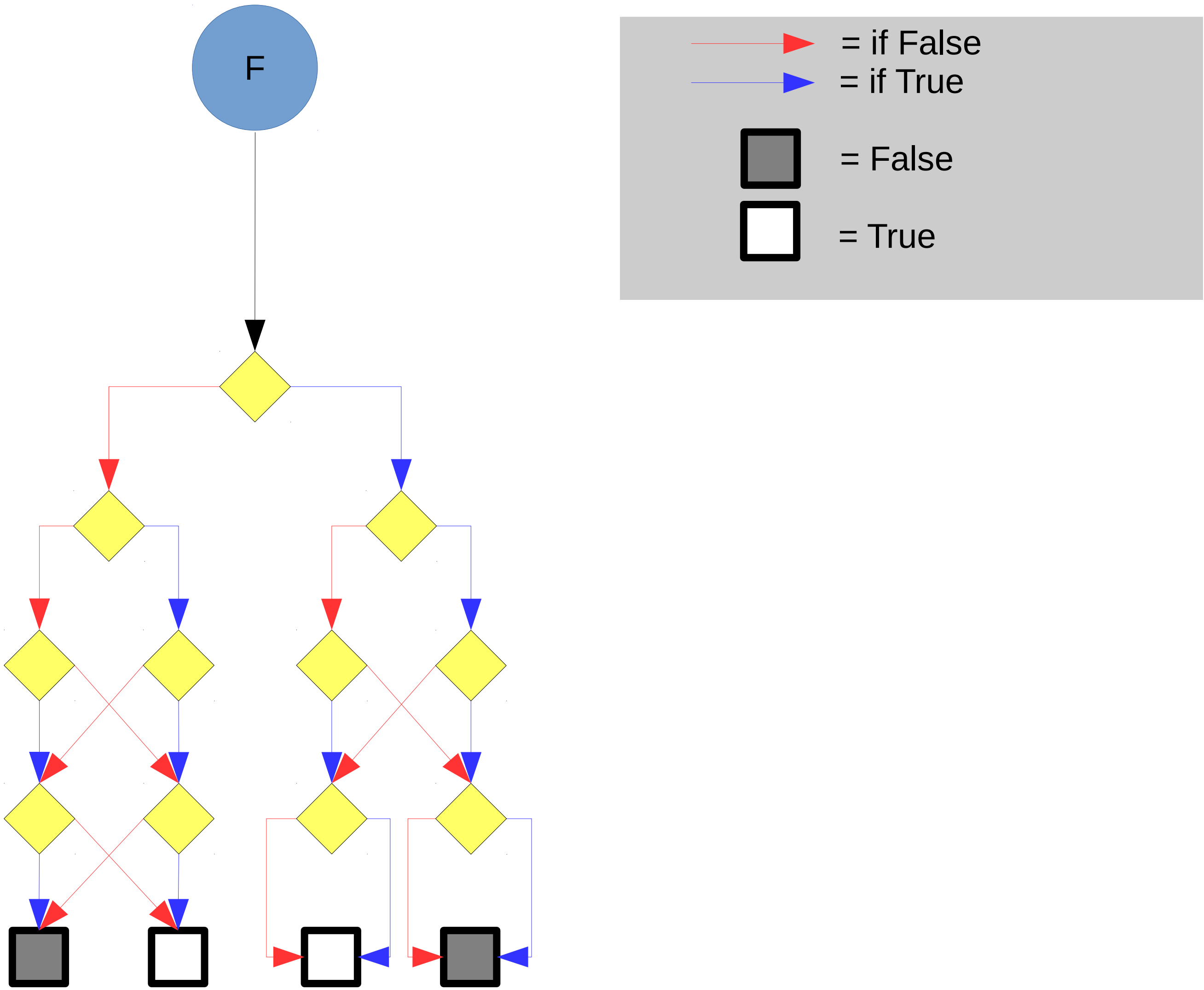
🟥 ➤ = if False

🟦 ➤ = if True

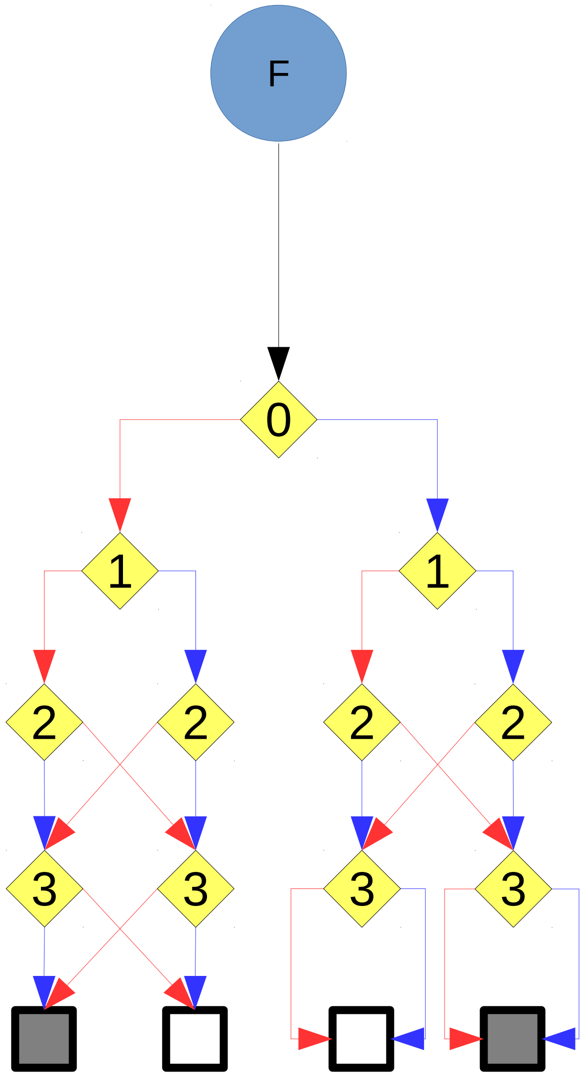
■ = False

□ = True

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



(Bryant) Step 2: we specify for each node: on which variable the decision is made



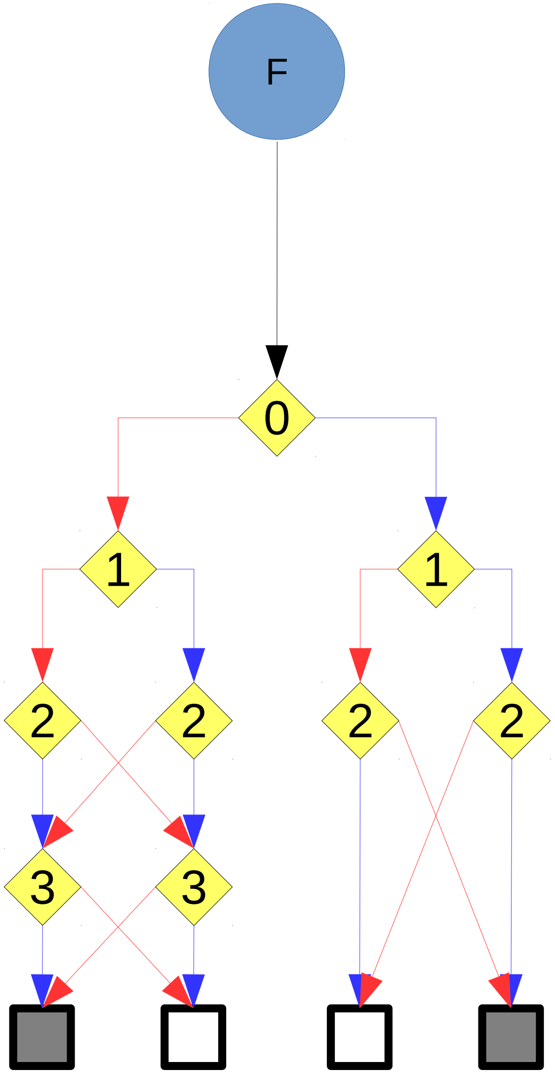
→ = if False

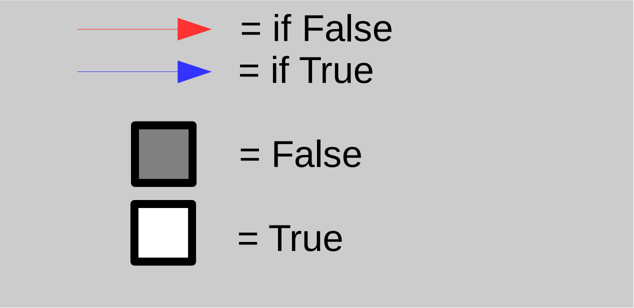
→ = if True

■ = False

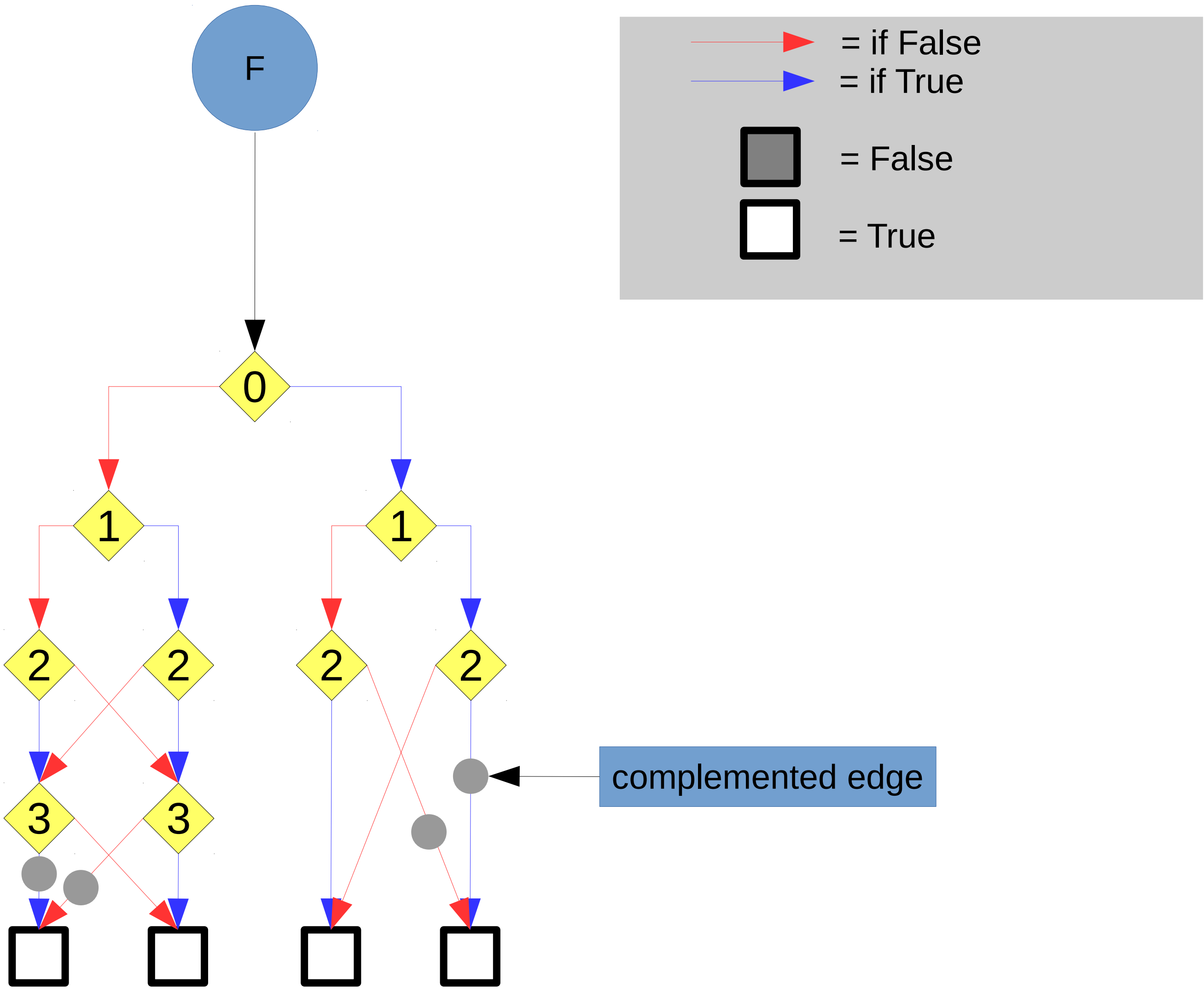
□ = True

(Bryant) Step 3: we remove useless decisions





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



= if False

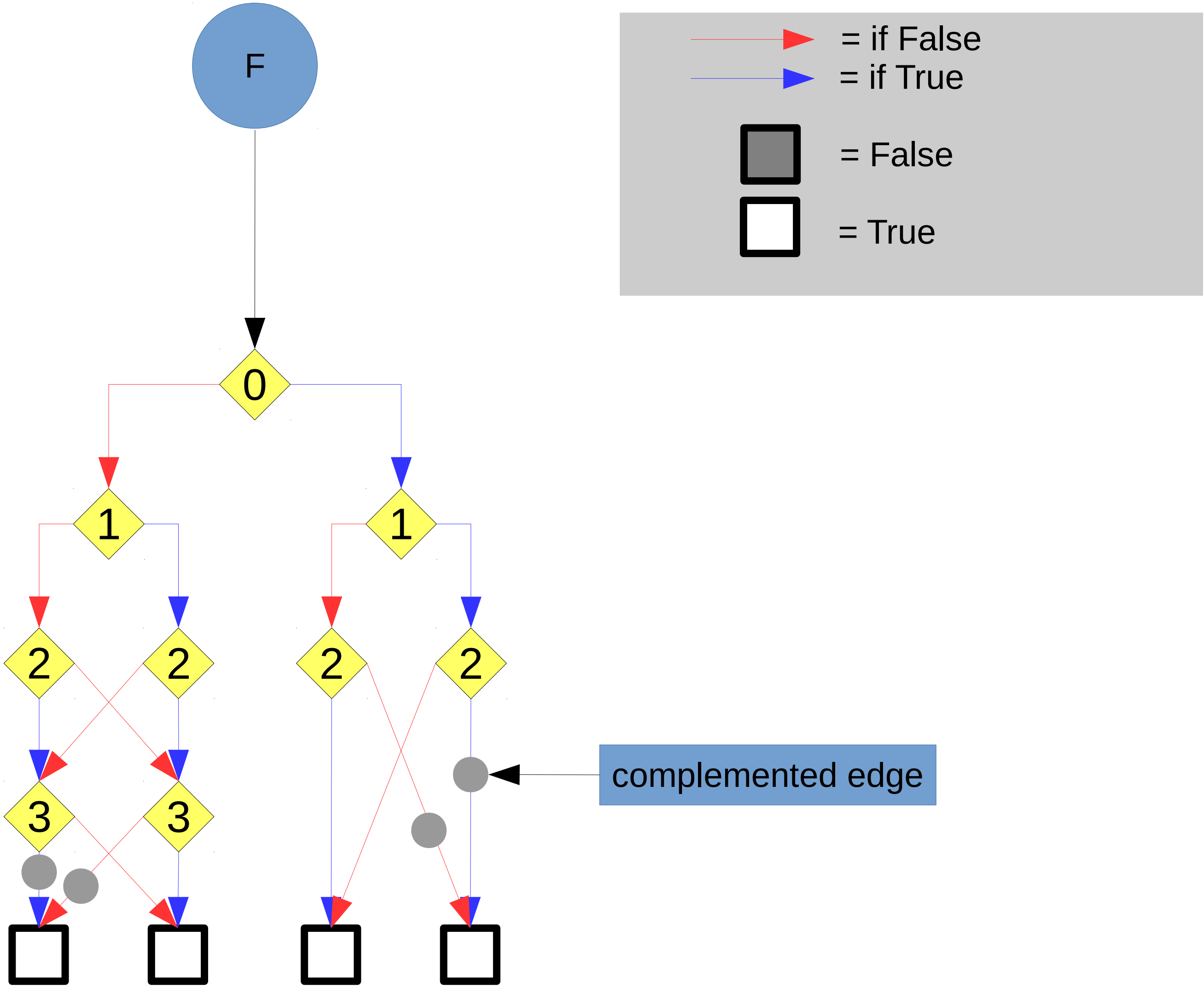
= if True

= False

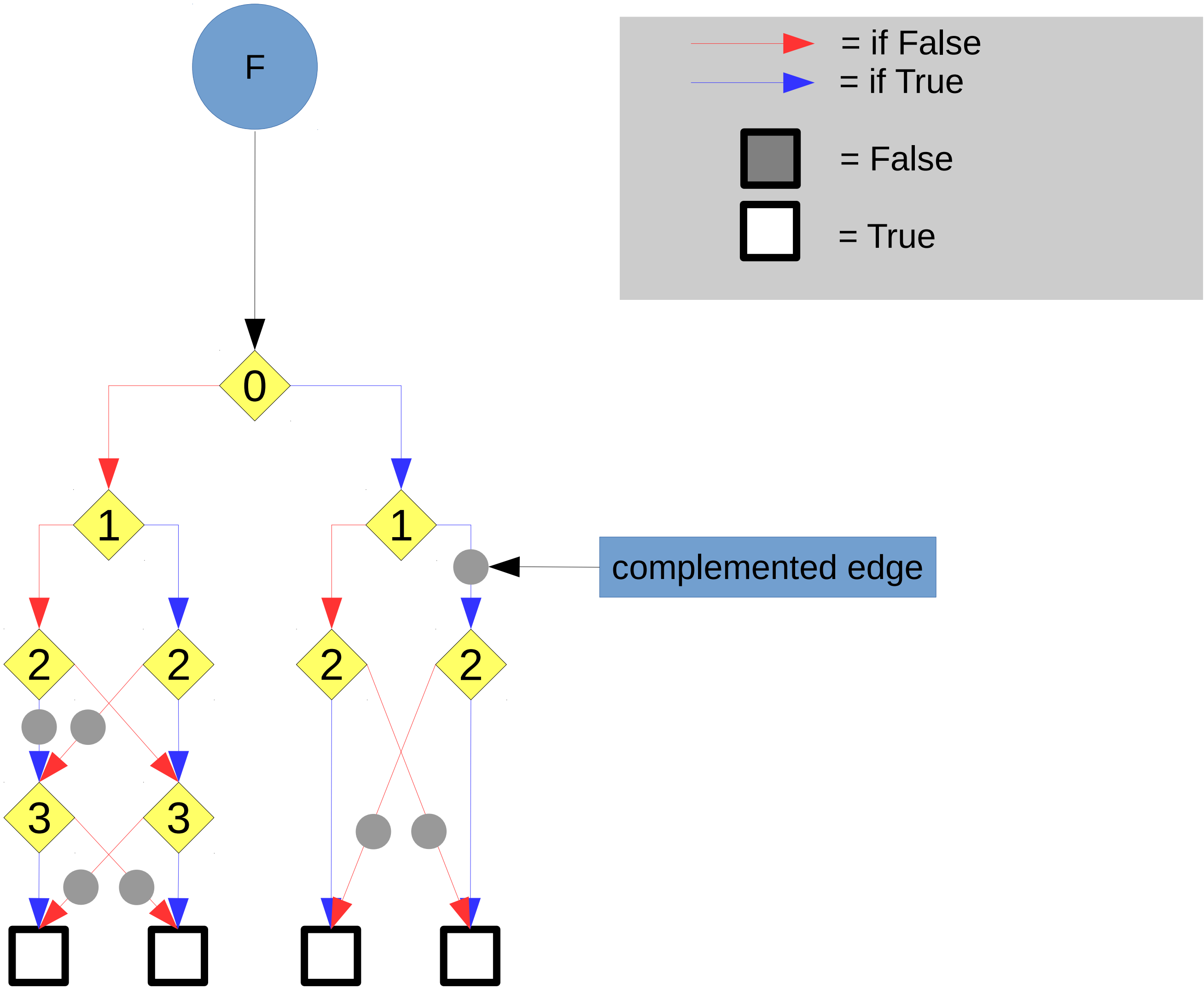
= True

complemented edge

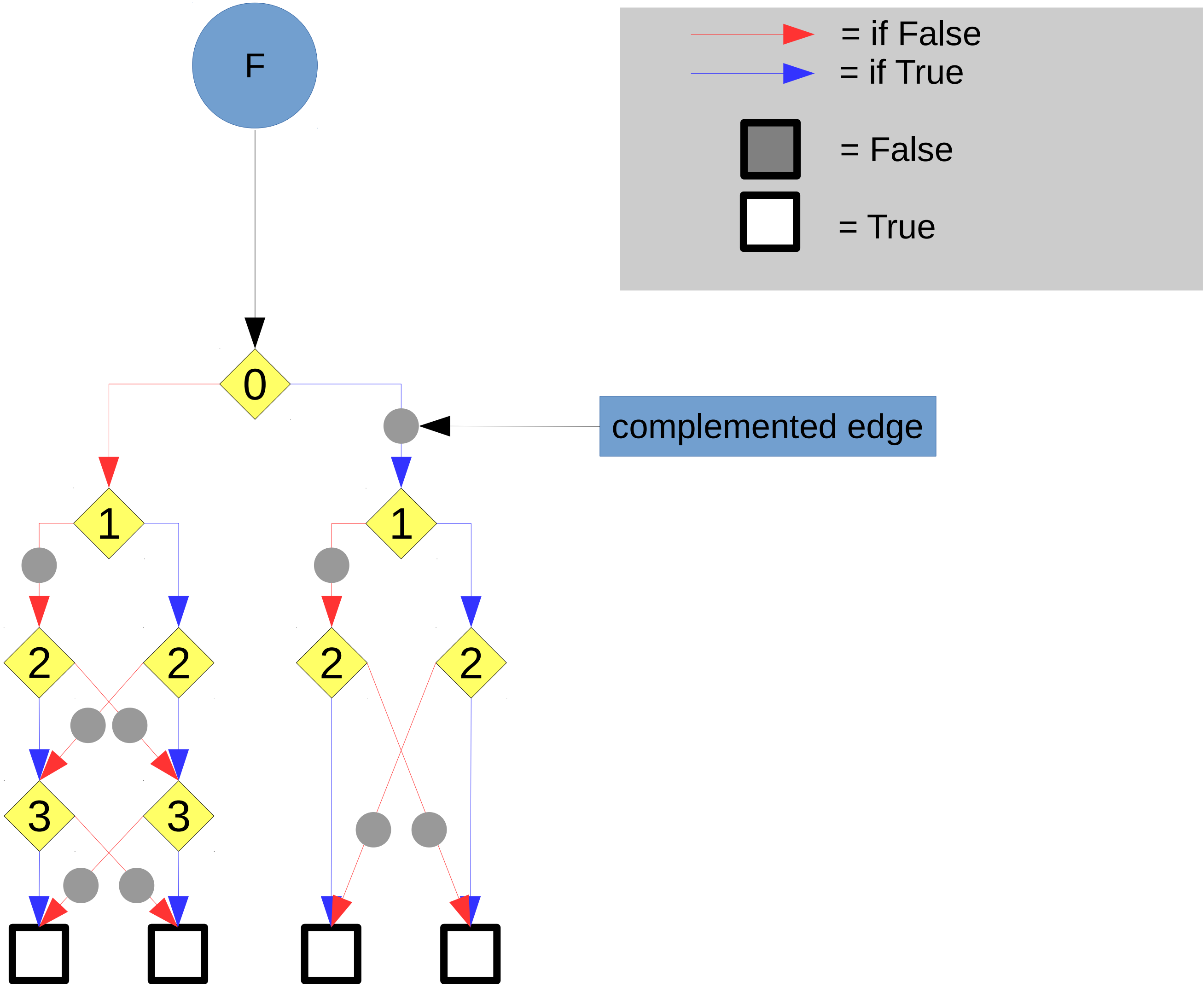
(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented



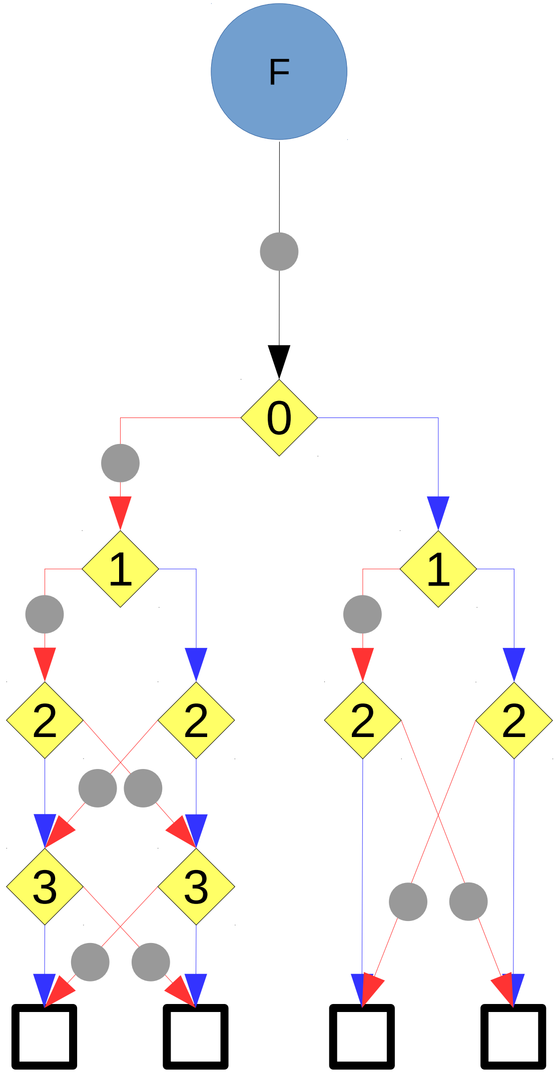
(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented

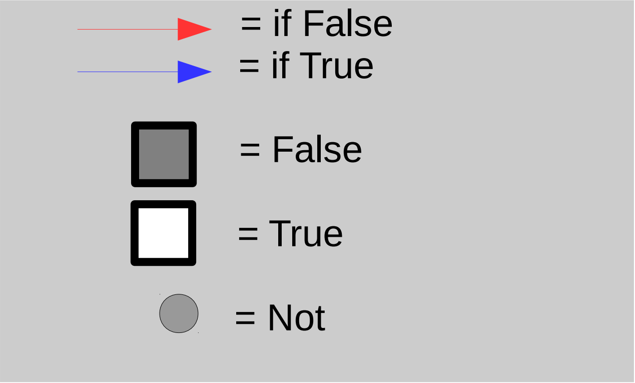


(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented

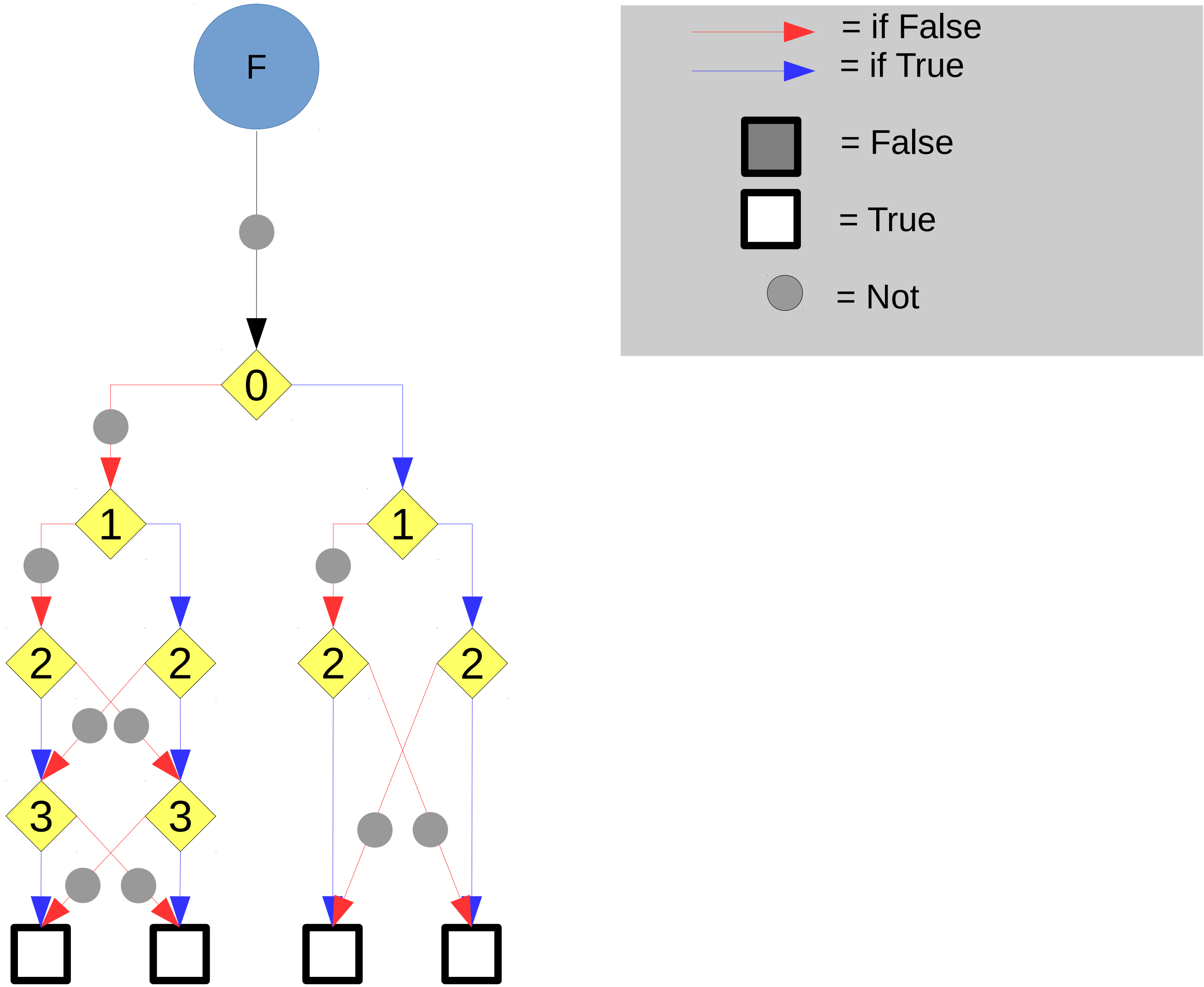


(Complemented Edges) Step 2 : we propagate inverted edges upward, ensuring that no “if True” edge is complemented

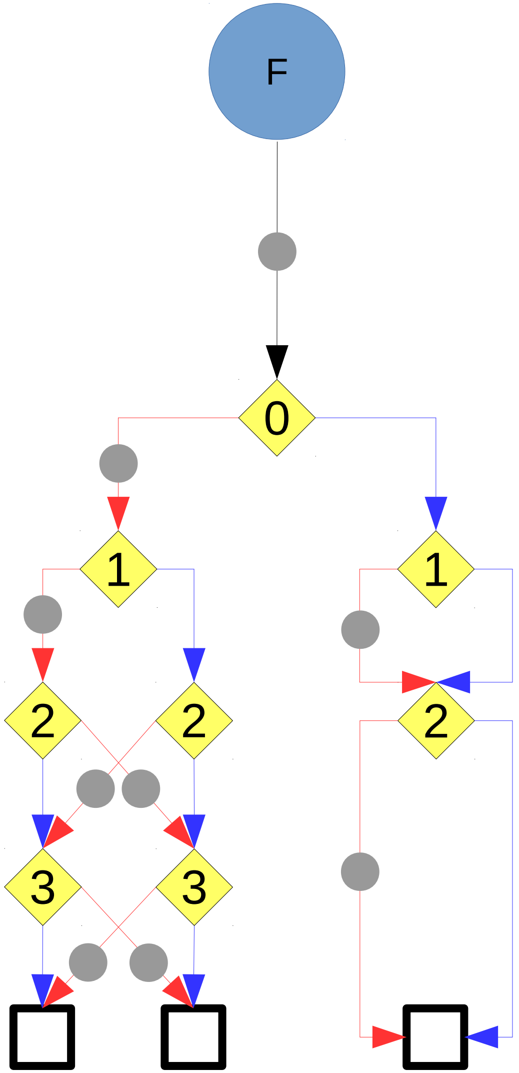




we merge isomorphic sub-graphs

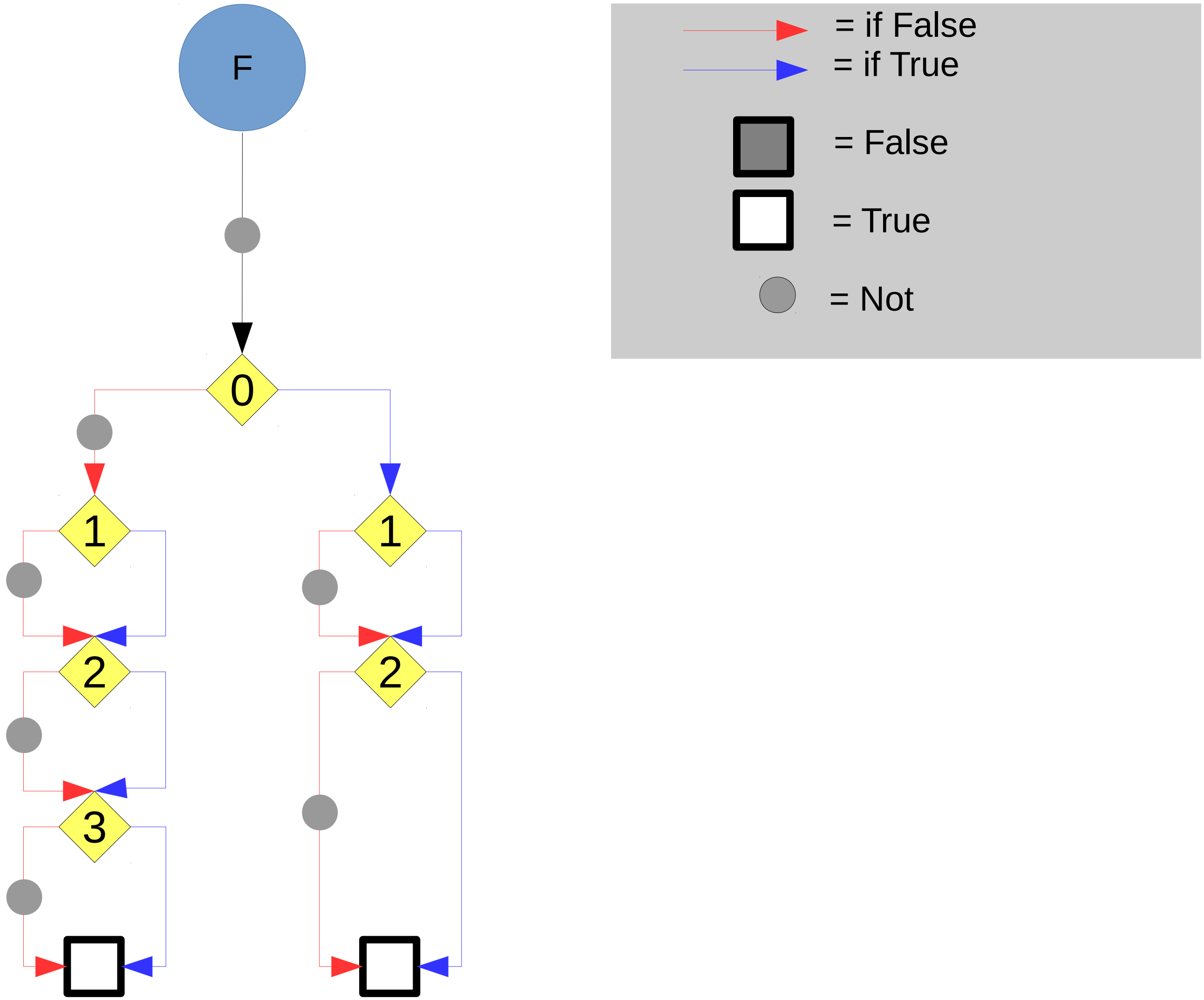


we merge isomorphic sub-graphs



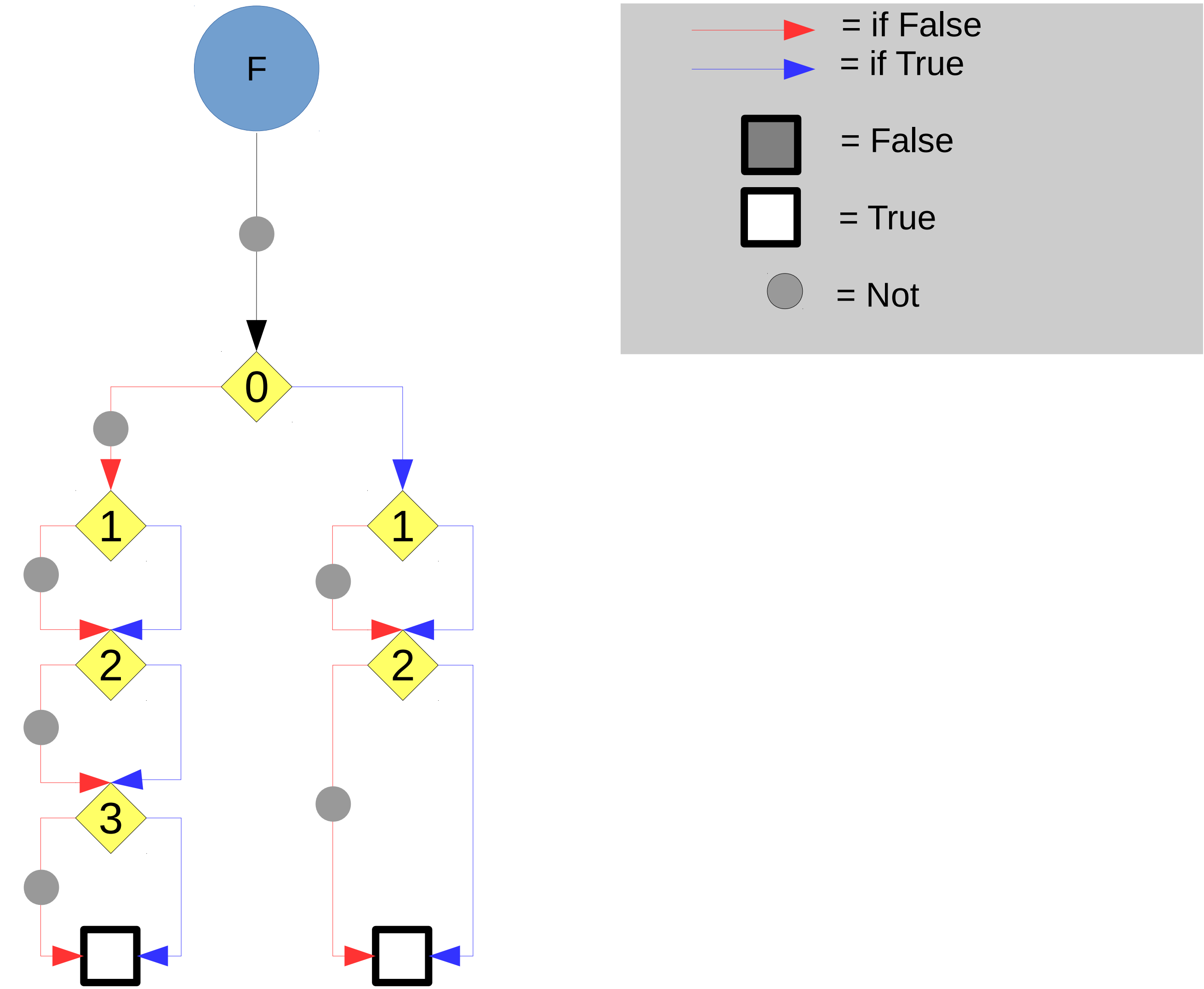
→ = if False
→ = if True
■ = False
□ = True
● = Not

we merge isomorphic sub-graphs



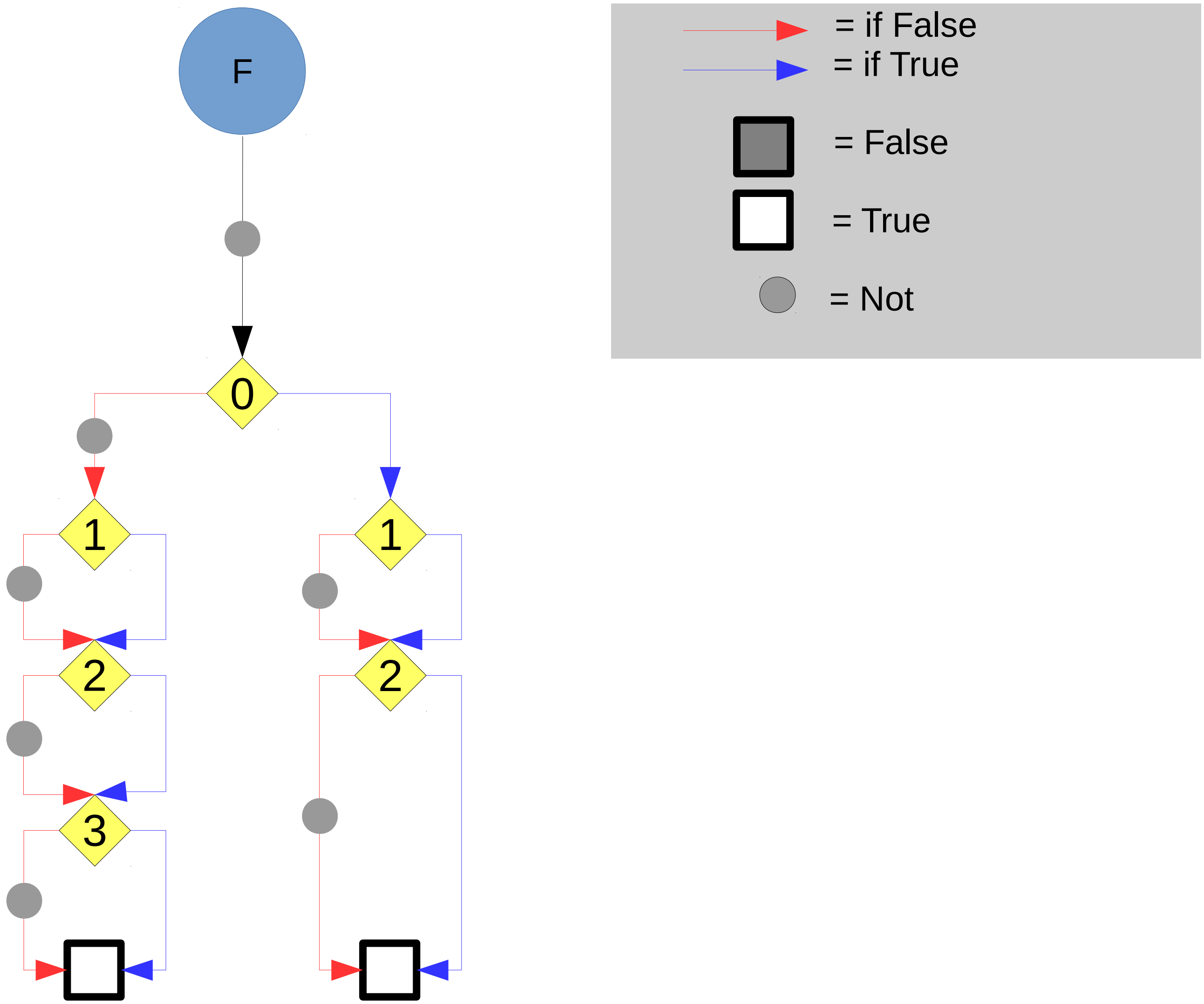
→ = if False
→ = if True
■ = False
□ = True
● = Not

we merge isomorphic sub-graphs



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

State Of The Art since 2000s

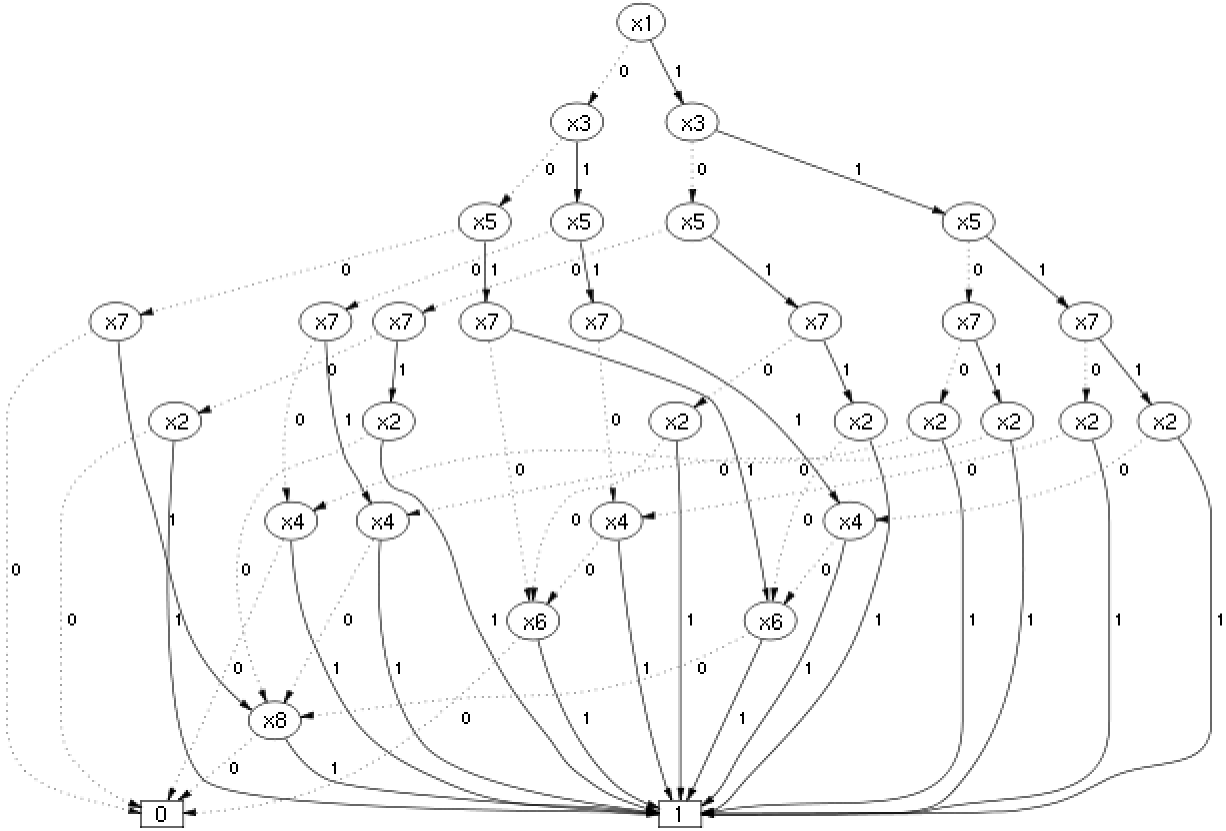


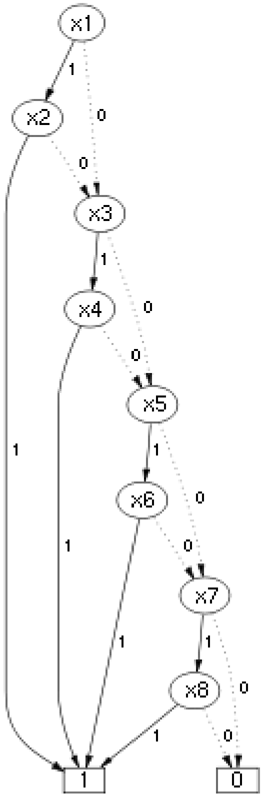
Reduced Ordered BDD

- =) :
  - 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

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee (x_5 \wedge x_6) \vee (x_7 \wedge x_8)$$

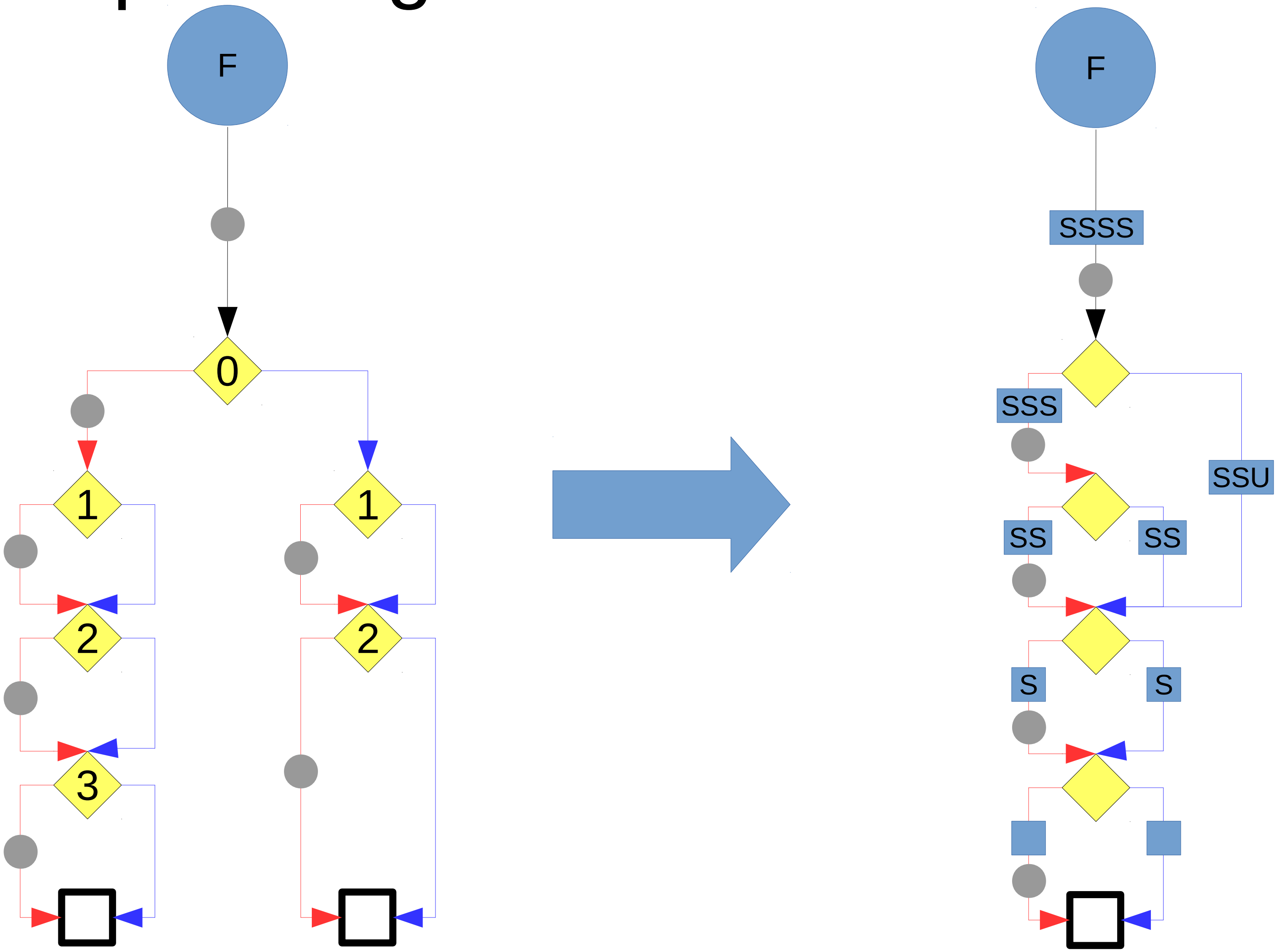




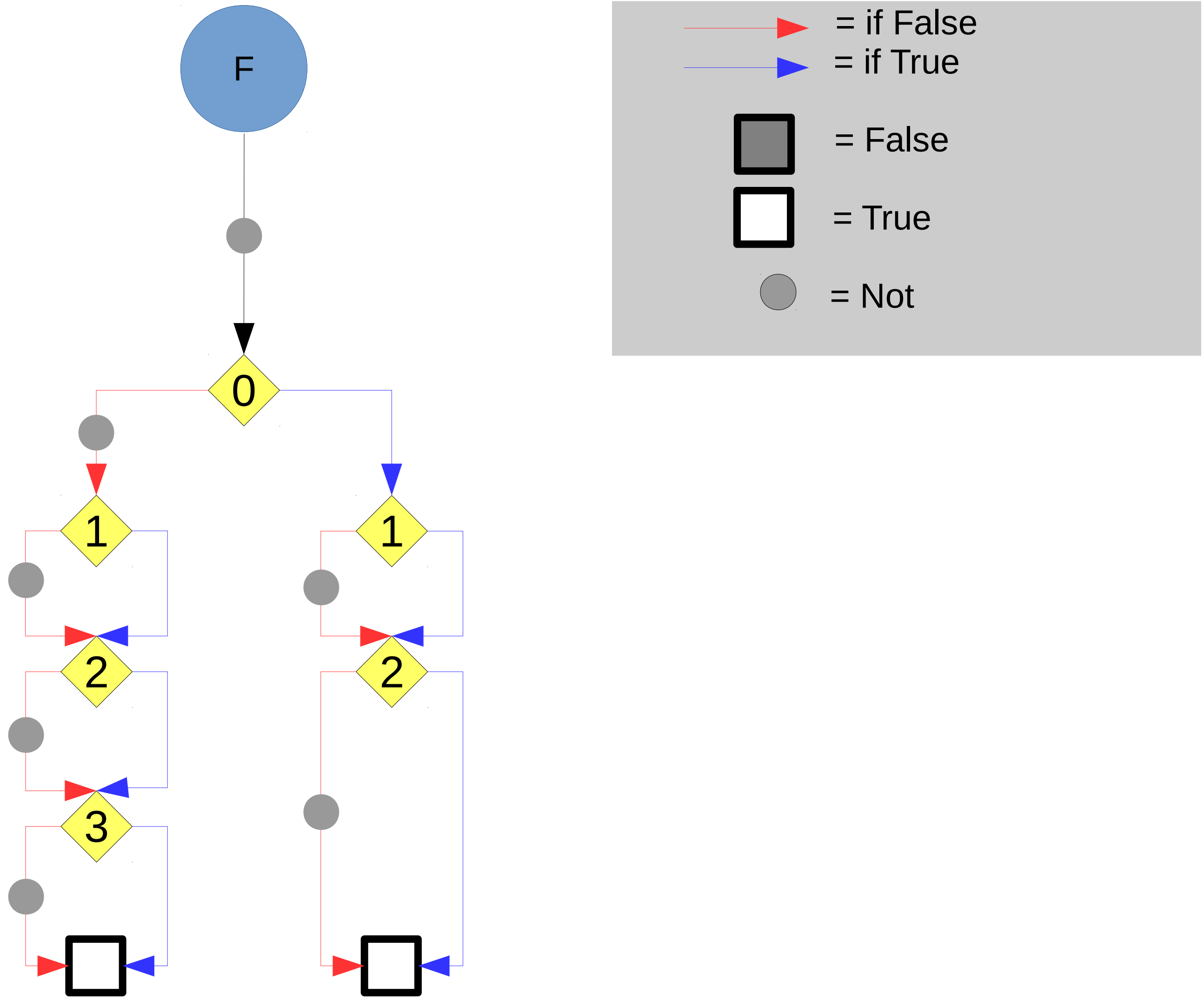
Objective

Reduce #node

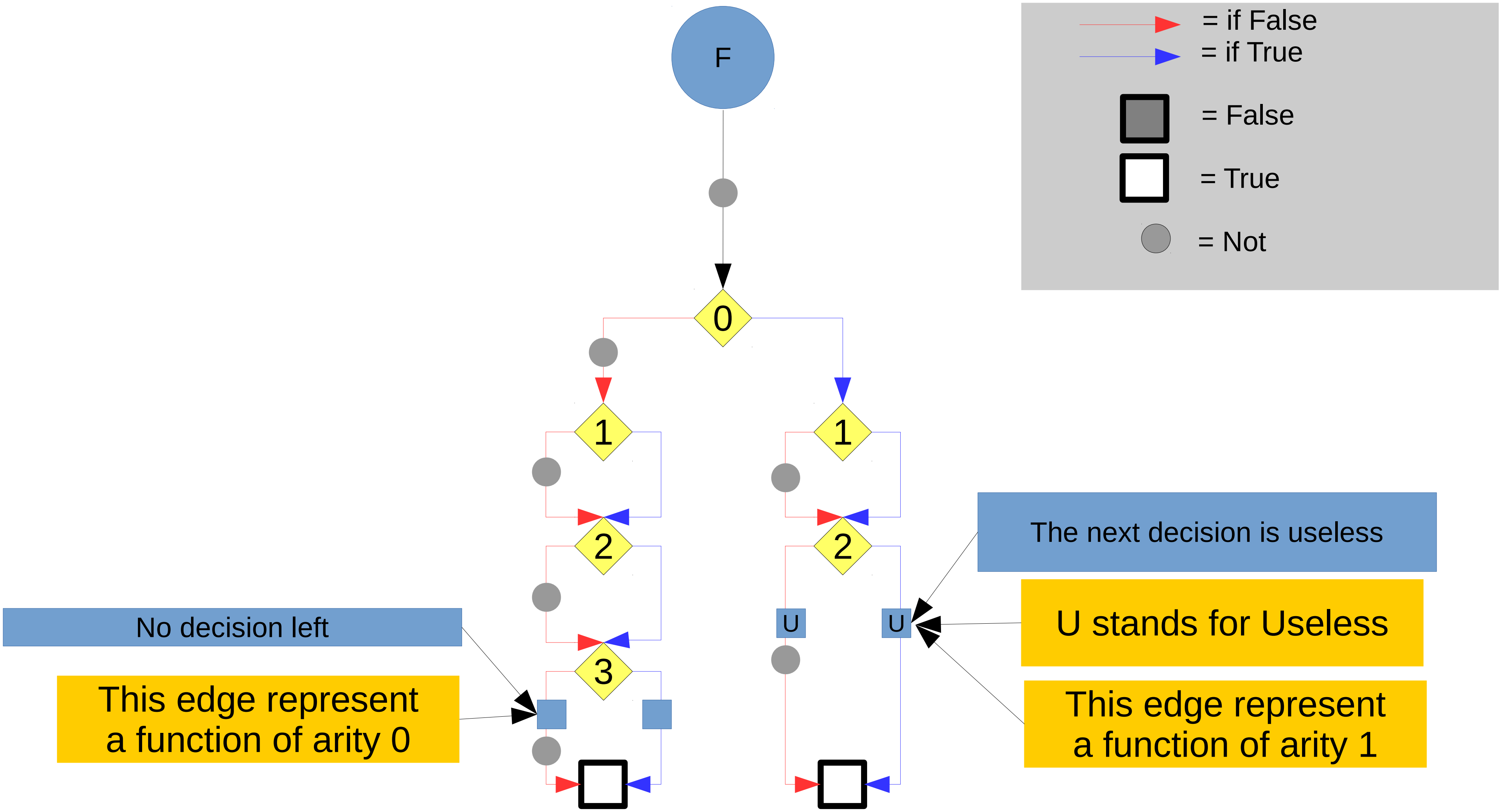
Section 2
Compressing a ROBDD into a GroBdd



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



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



No decision left

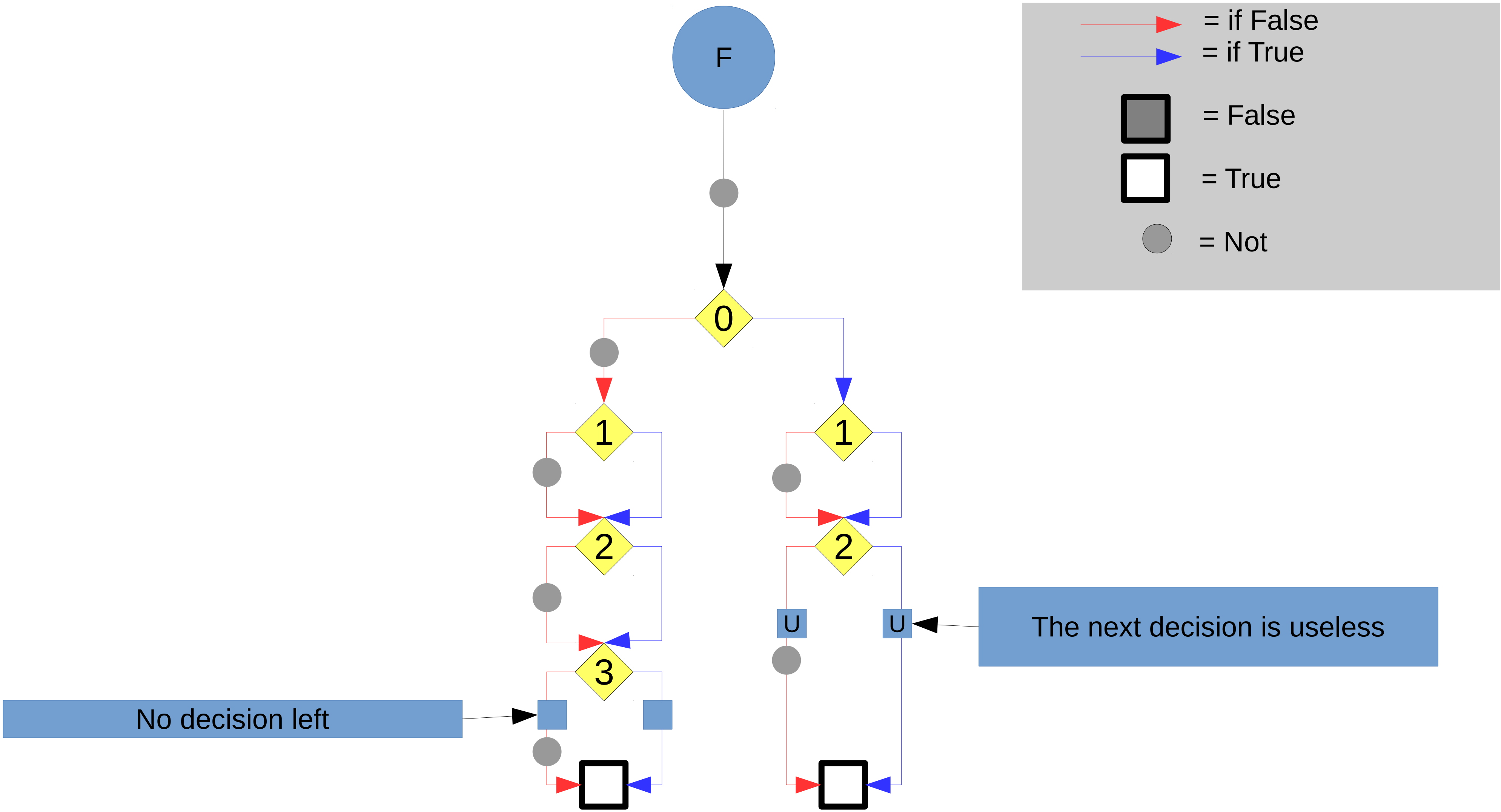
This edge represent a function of arity 0

The next decision is useless

U stands for Useless

This edge represent a function of arity 1

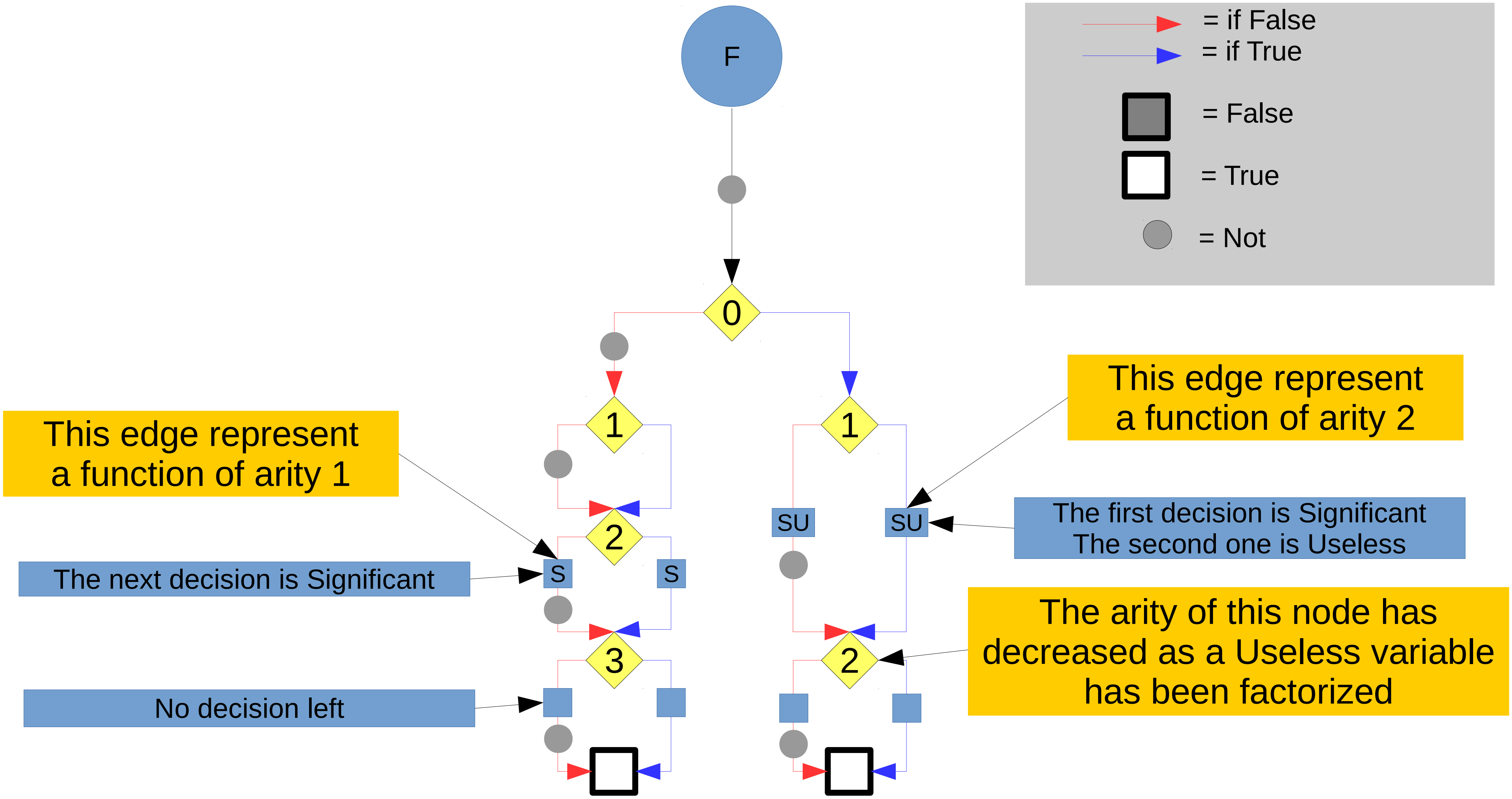
(Model 1) Step 2: we factorize useless variables



No decision left

The next decision is useless

(Model 1) Step 2: we factorize useless variables



→ = if False

→ = if True

■ = False

□ = True

● = Not

This edge represent
a function of arity 1

The next decision is Significant

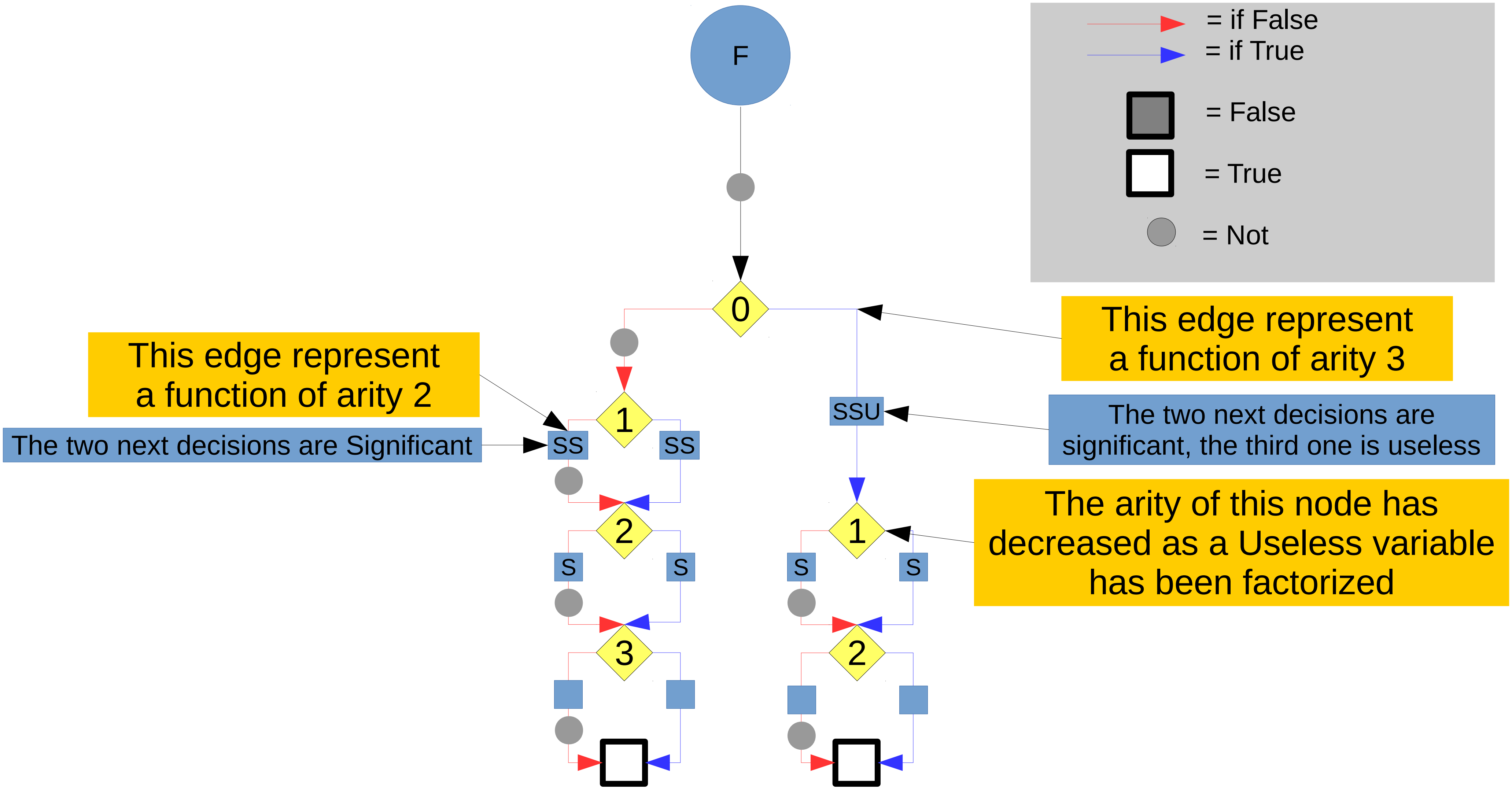
No decision left

This edge represent
a function of arity 2

The first decision is Significant
The second one is Useless

The arity of this node has
decreased as a Useless variable
has been factorized

(Model 1) Step 2: we factorize useless variables



This edge represent a function of arity 2

The two next decisions are Significant

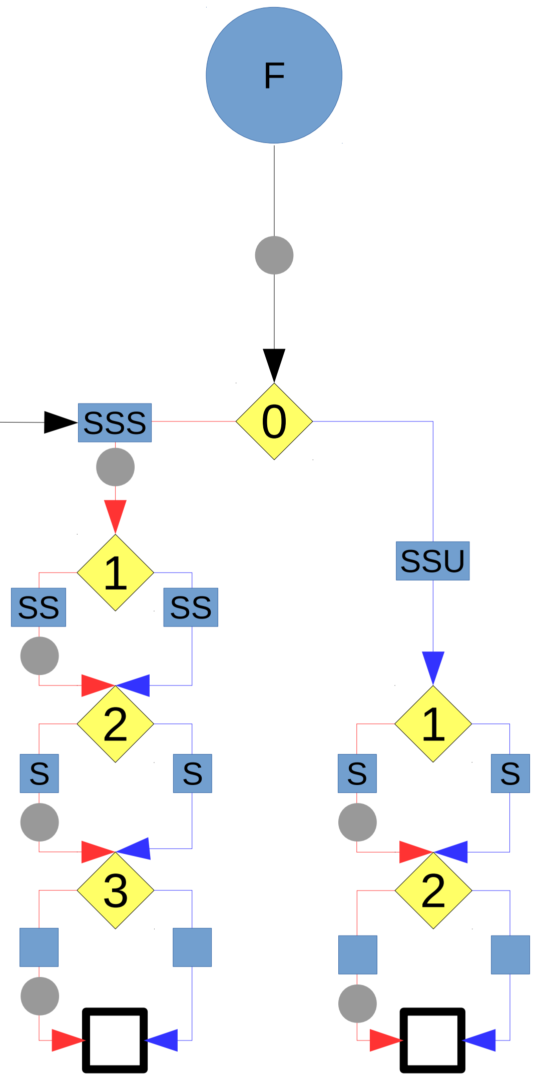
This edge represent a function of arity 3

The two next decisions are significant, the third one is useless

The arity of this node has decreased as a Useless variable has been factorized

(Model 1) Step 2: we factorize useless variables

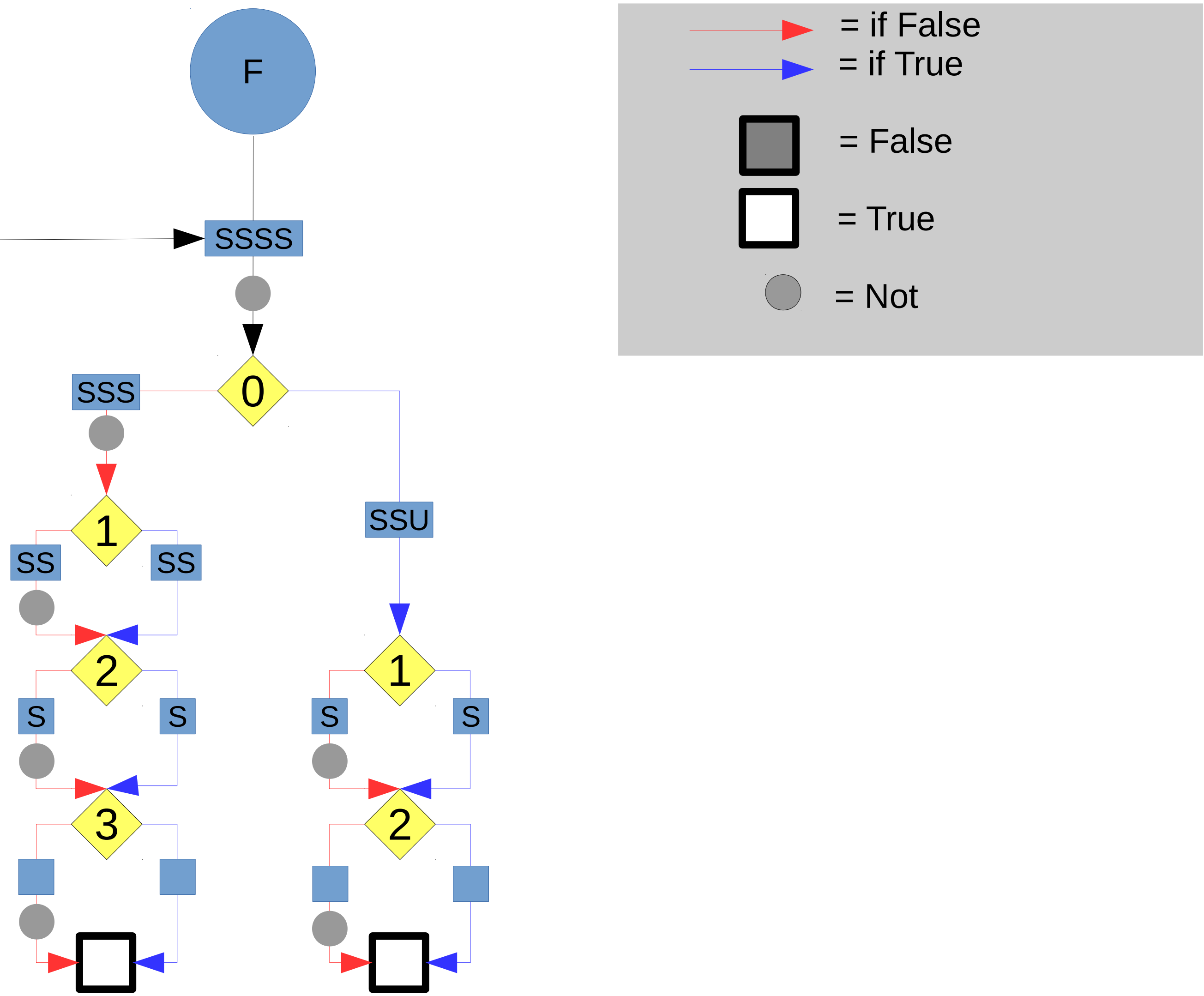
The three next decisions are significant



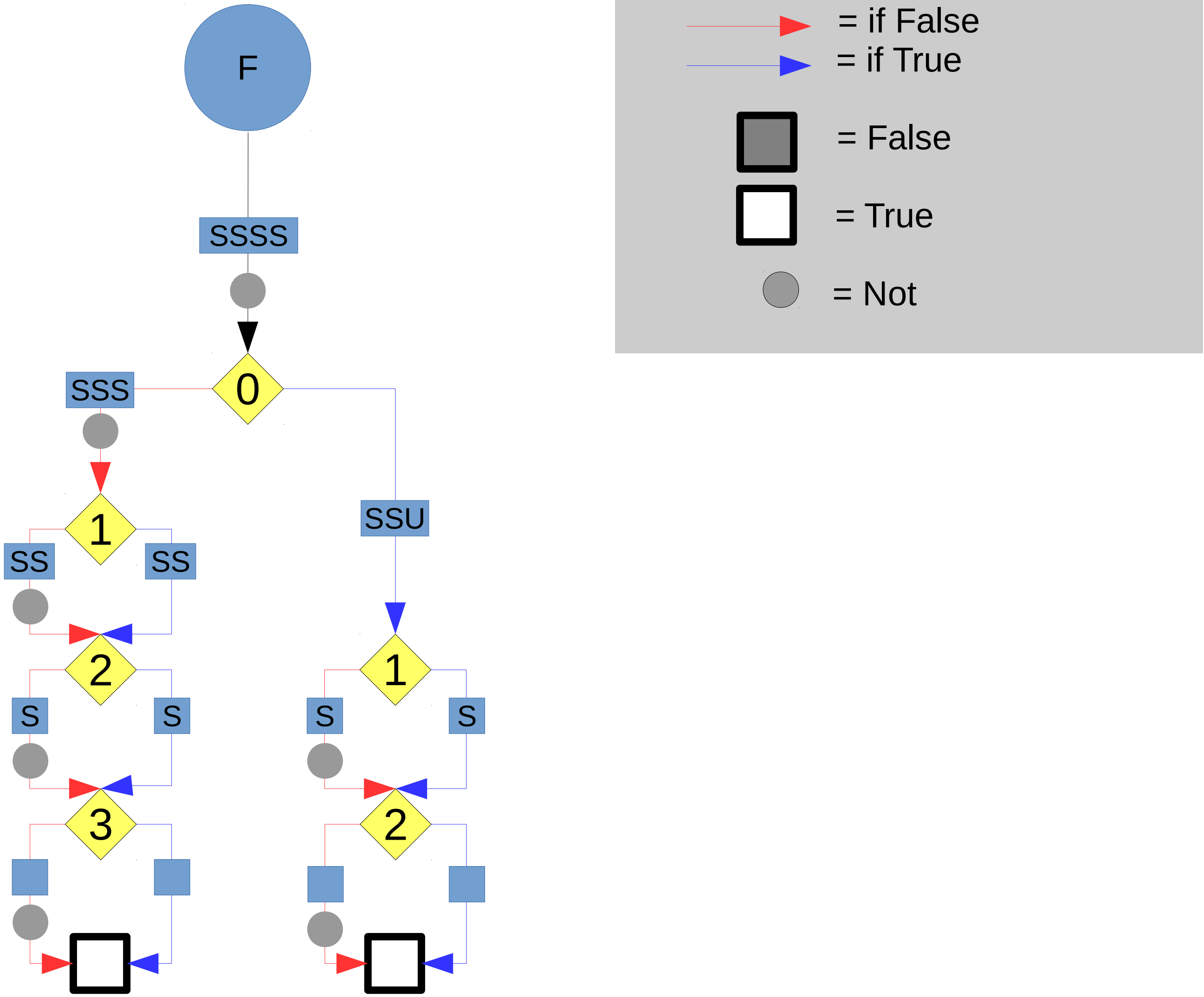
- ➜ = if False
- ➜ = if True
- ■ = False
- □ = True
- ● = Not

(Model 1) Step 2: we factorize useless variables

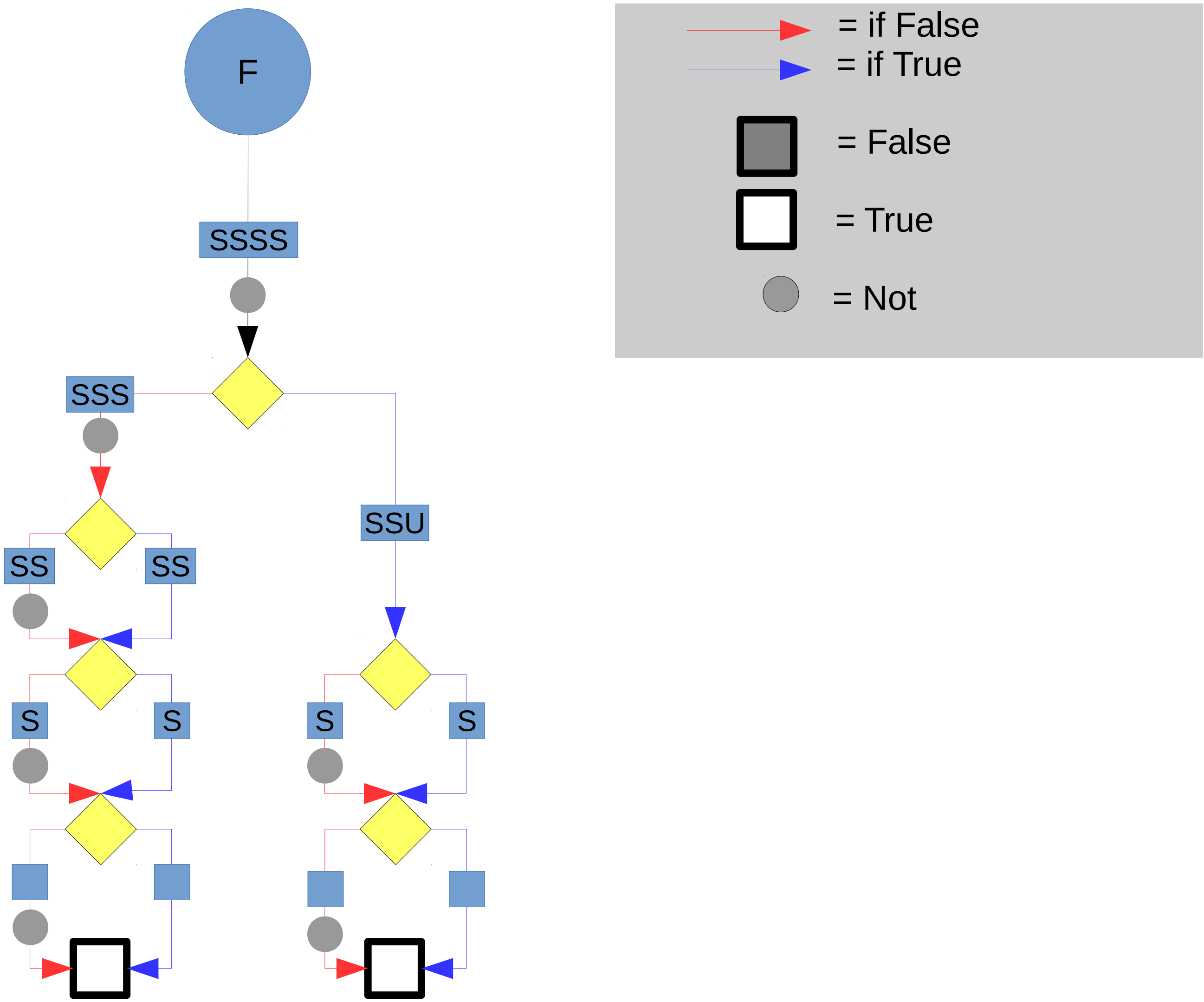
The four next decisions are significant



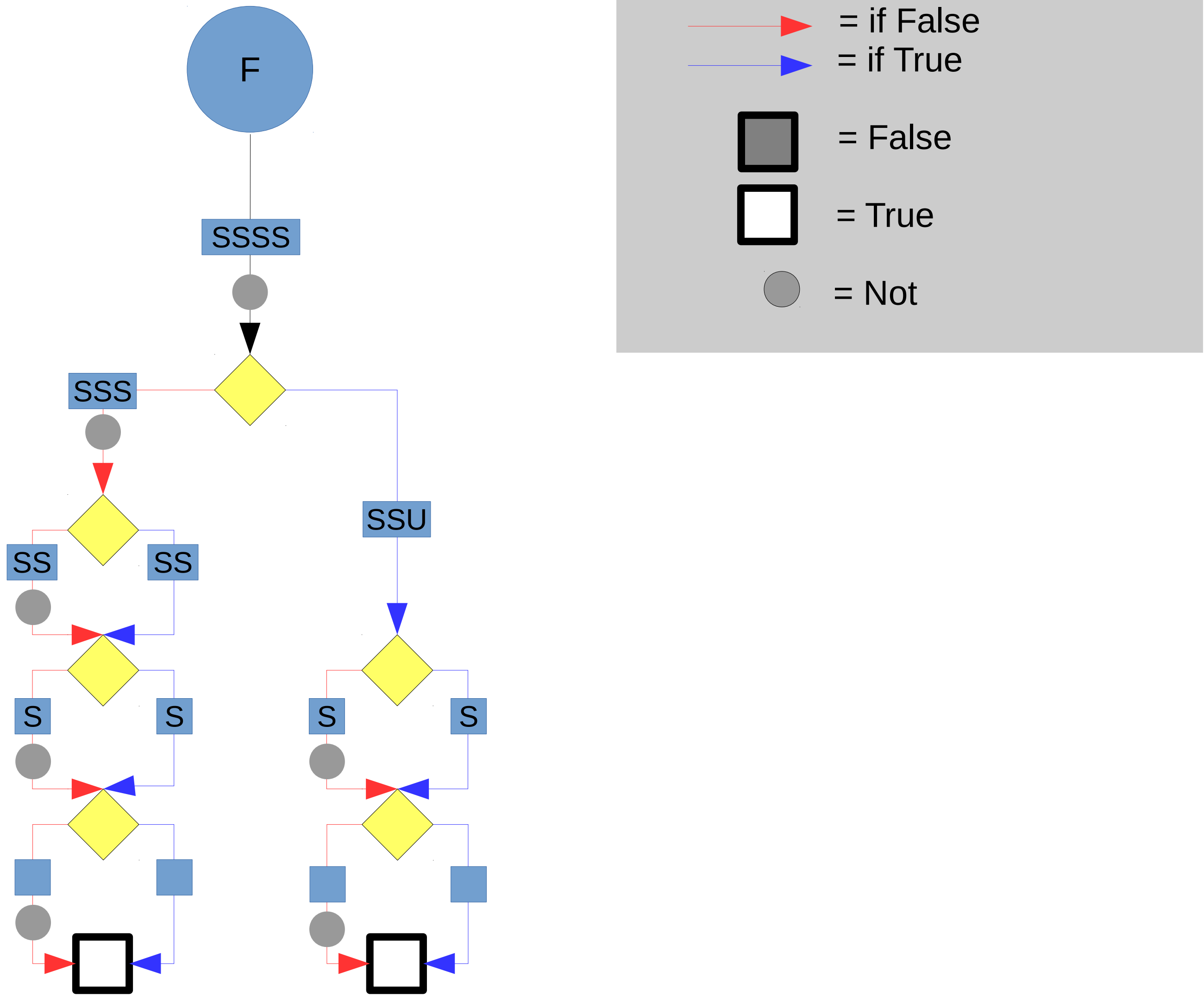
(Model 1) Step 3: we forget every node’s depth



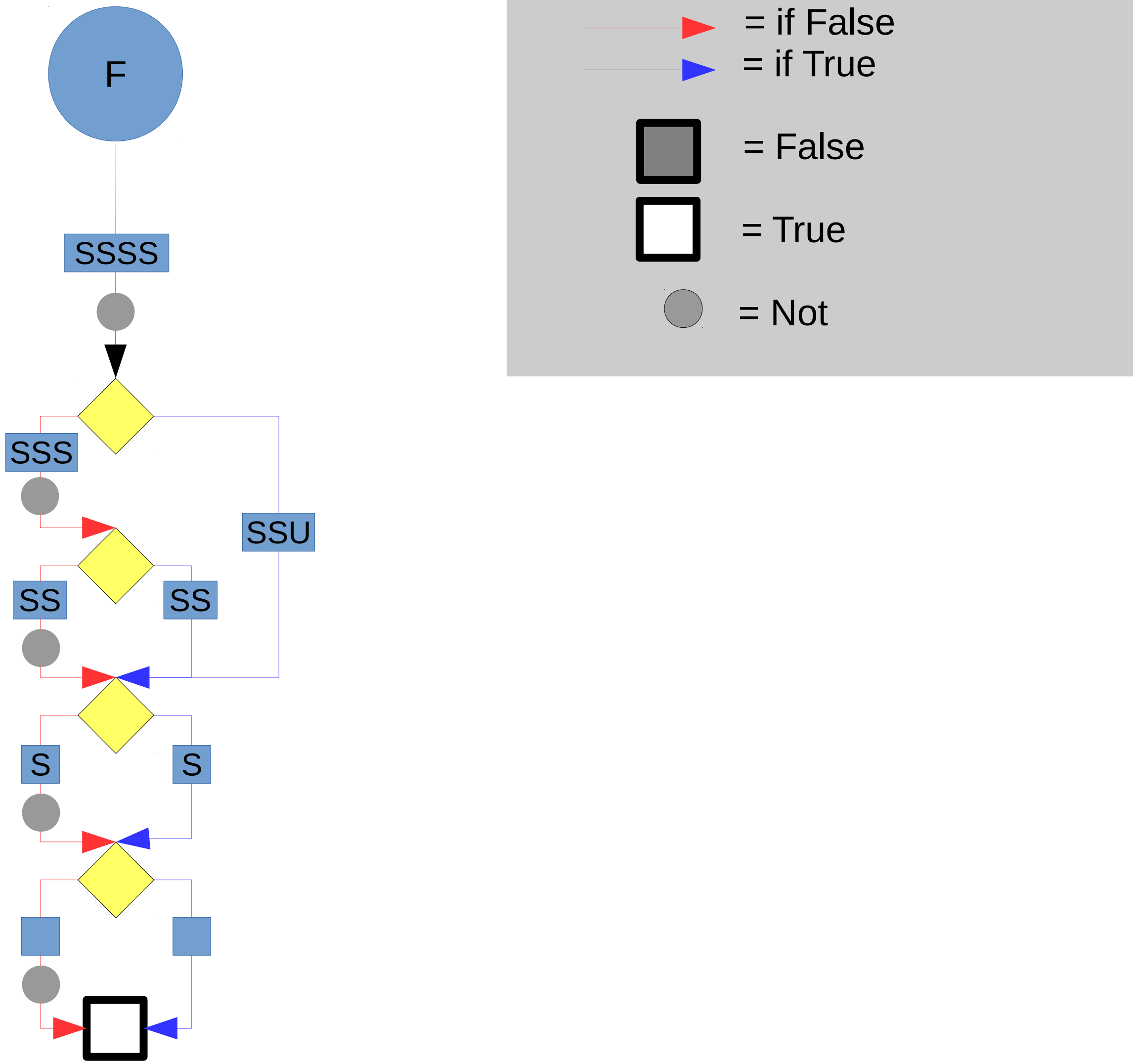
(Model 1) Step 3: we forget every node’s decision variable



we merge isomorphic sub-graphs



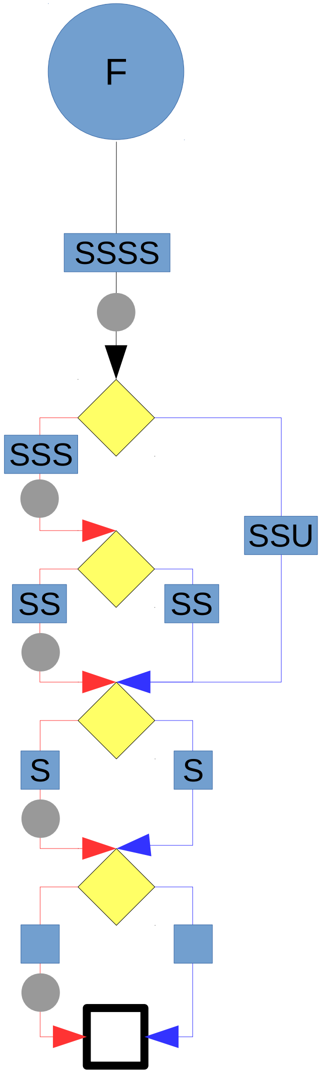
we merge isomorphic sub-graphs



Section 3
Compiling a formula into a GroBdd

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

Represents a vector of four elements:
$x_0^3=\left(x_{0,}\, x_{1,}\, x_{2,}\, x_3\right)$



How to compile a formula into a GroBdd

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

How to compile a formula into a GroBdd

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

→ = if False

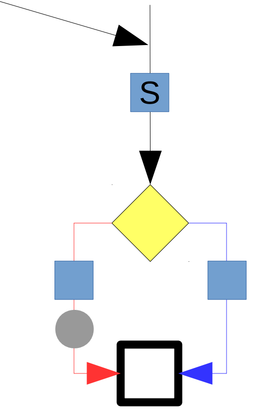
→ = if True

■ = False

□ = True

● = Not

Step 1: we build the identity function



How to compile a formula into a GroBdd

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

→ = if False

→ = if True

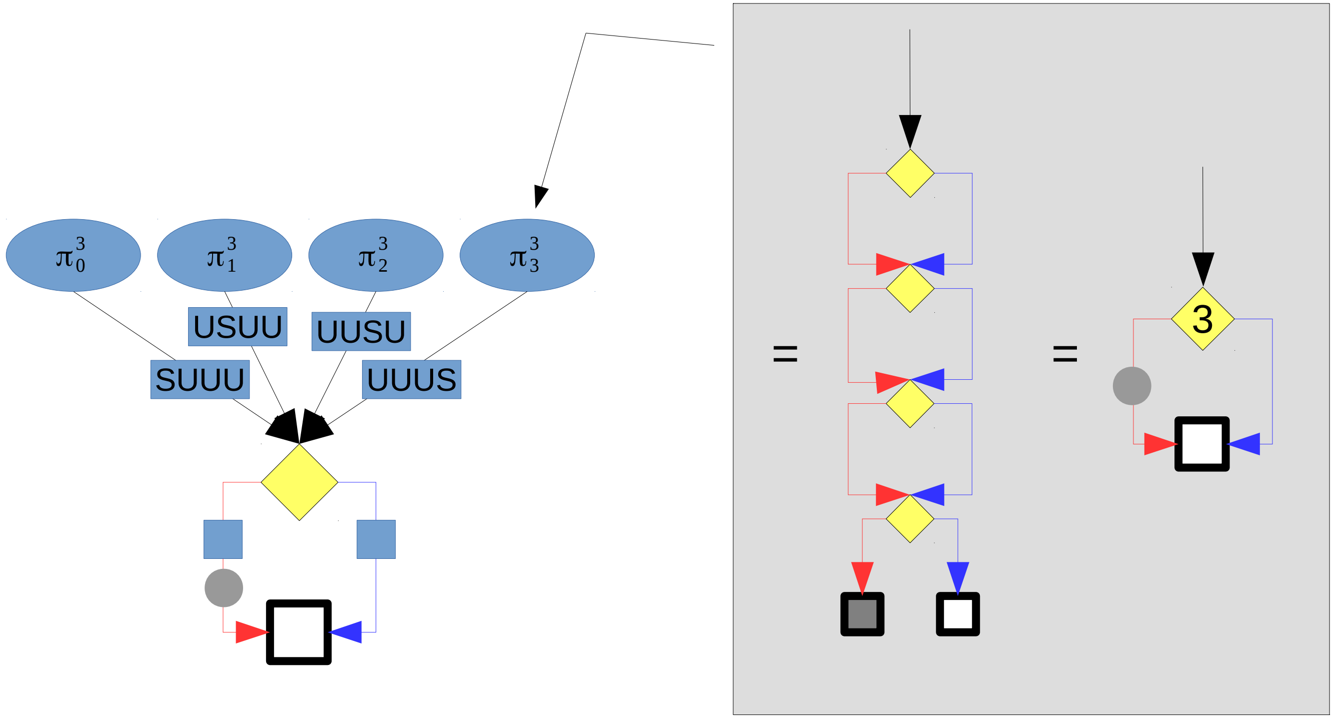
■ = False

□ = True

● = Not

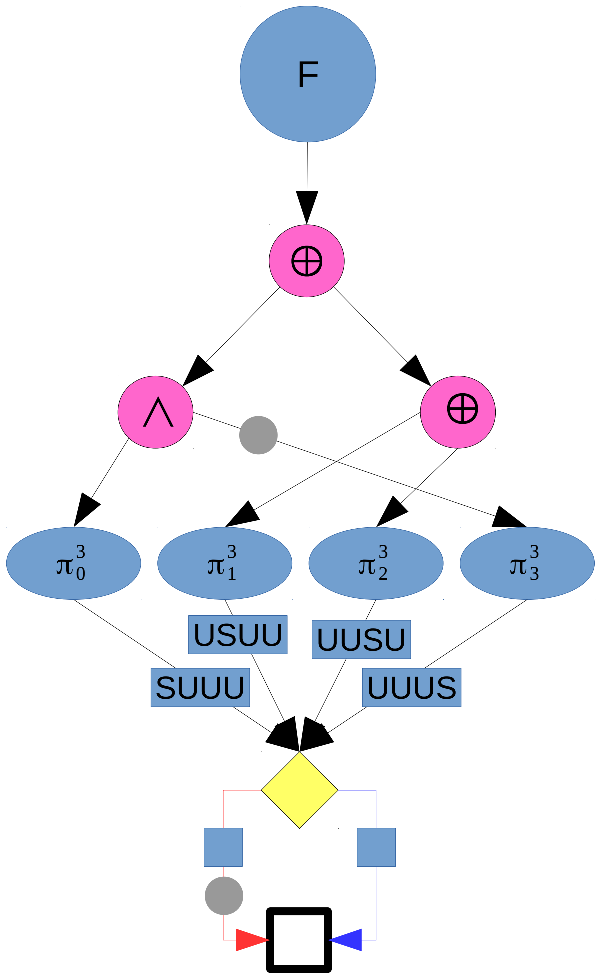
Step 2: we build one projection per variable

$\pi_k^n\left(x_0^n\right)=x_k$



How to compile a formula into a GroBdd

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



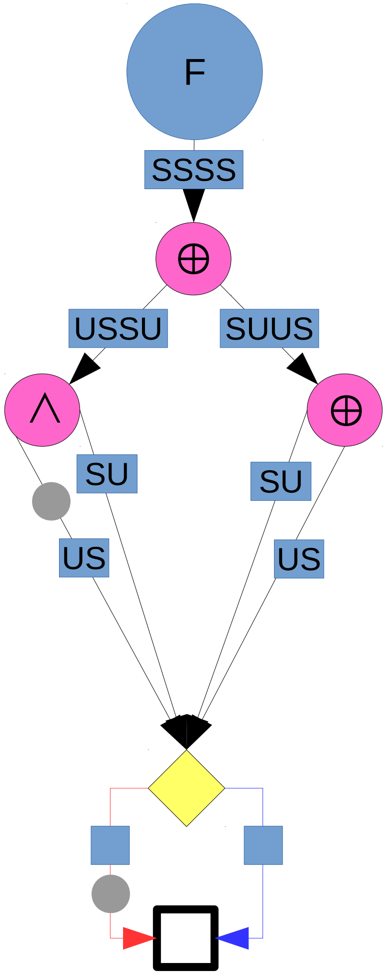
→ = if False
→ = if True
■ = False
□ = True
● = Not

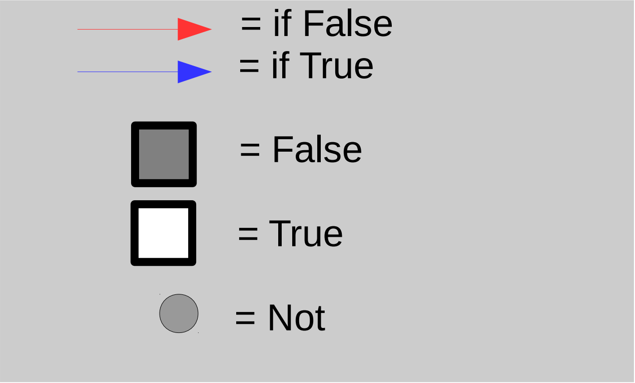
∧ ⊕ ◆ Virtual nodes

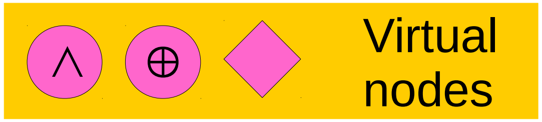
Step 3: we build the formula

How to compile a formula into a GroBdd

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





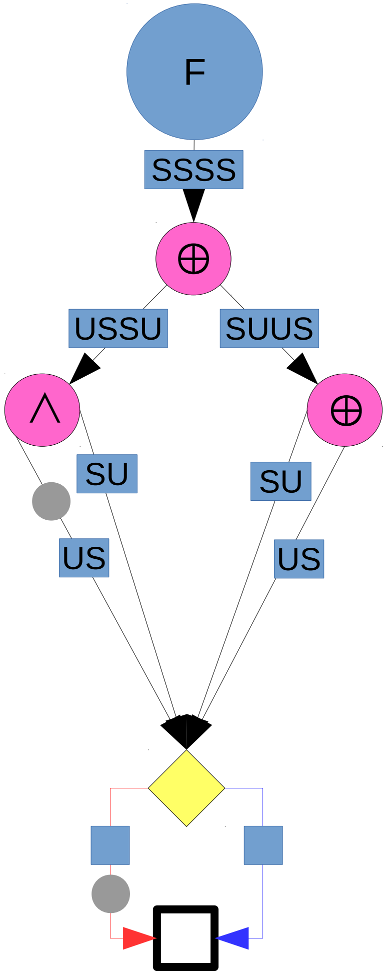


Virtual nodes

Step 4: we factorize useless variables

How to compile a formula into a GroBdd

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



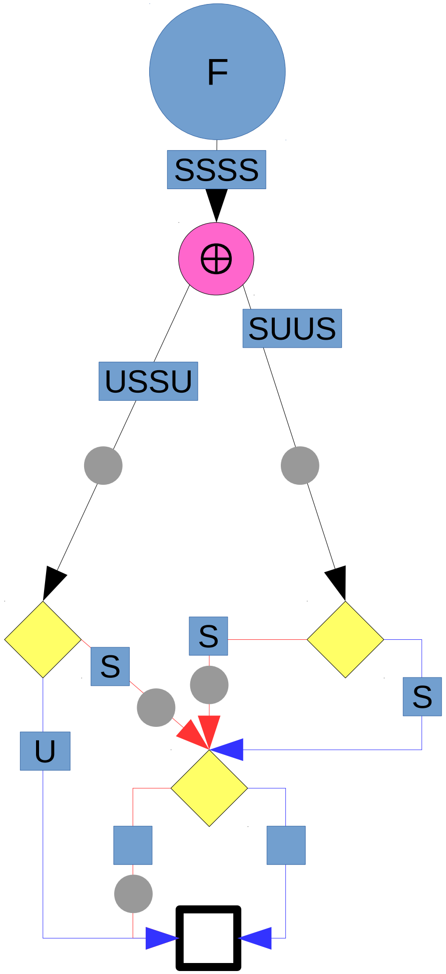
- 🟥→ = if False
- 🟦→ = if True
- ■ = False
- □ = True
- ● = Not

∧ ⊕ ◆ Virtual nodes

Step 5: we compute operator nodes

How to compile a formula into a GroBdd

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



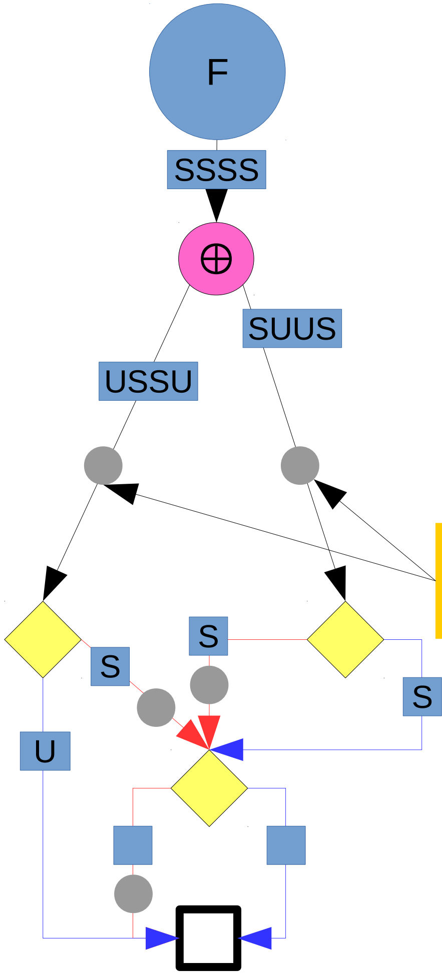
🟥→ = if False
🟦→ = if True
■ = False
□ = True
● = Not

∧ ⊕ ◆ Virtual nodes

Step 5: we compute operator nodes

How to compile a formula into a GroBdd

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



🟥→ = if False
🟦→ = if True
■ = False
□ = True
● = Not

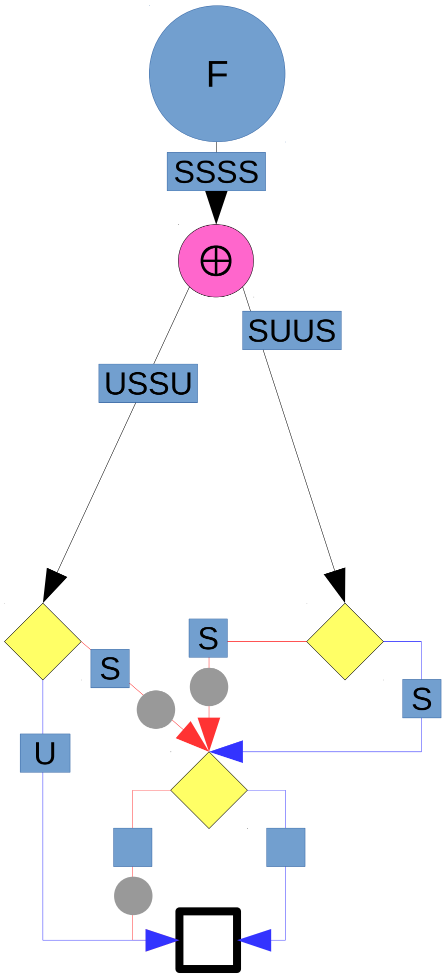
∧ ⊕ ◆ Virtual nodes

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

Step 5: we compute operator nodes

How to compile a formula into a GroBdd

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



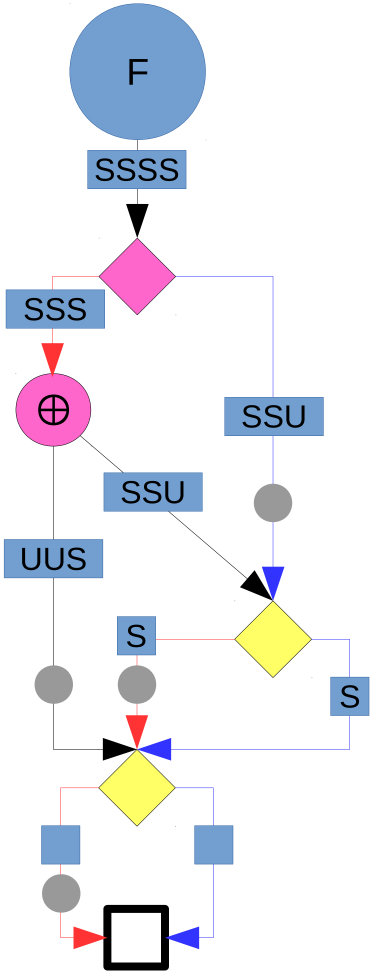
→ = if False
→ = if True
■ = False
□ = True
● = Not

∧ ⊕ ◆ Virtual nodes

Step 5: we compute operator nodes

How to compile a formula into a GroBdd

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



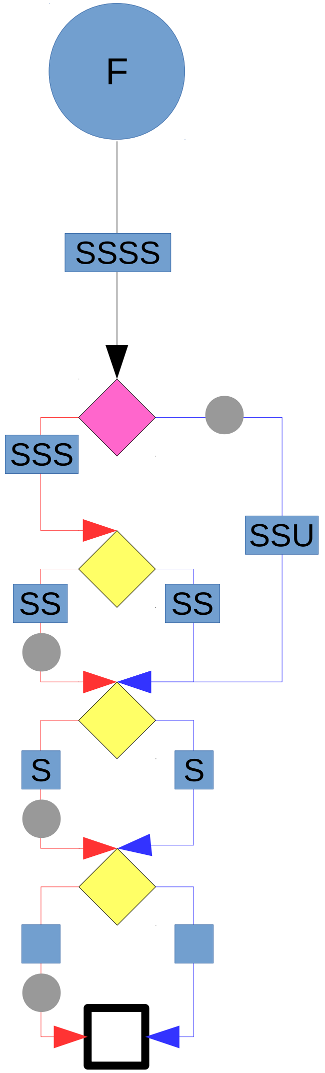
→ = if False
→ = if True
■ = False
□ = True
● = Not

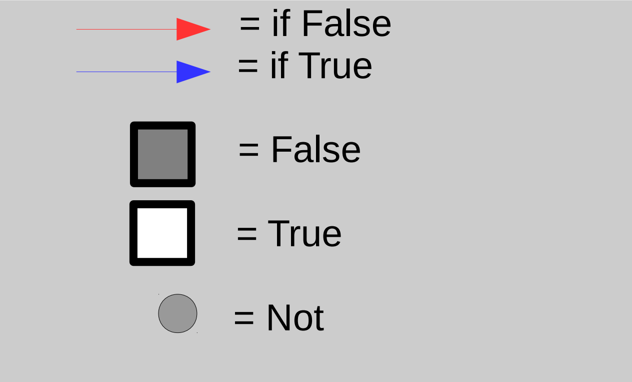
∧ ⊕ ◆ Virtual nodes

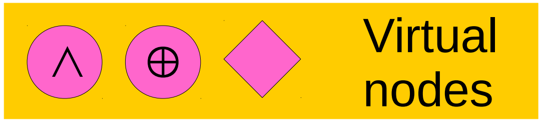
Step 5: we compute operator nodes

How to compile a formula into a GroBdd

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



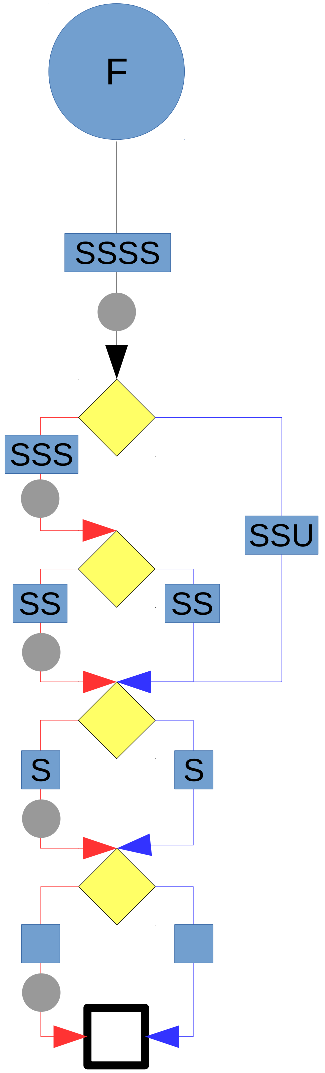




Step 5: we compute operator nodes

How to compile a formula into a GroBdd

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



→ = if False
→ = if True
■ = False
□ = True
● = Not

∧ ⊕ ◆ Virtual nodes

Step 5: we compute operator nodes

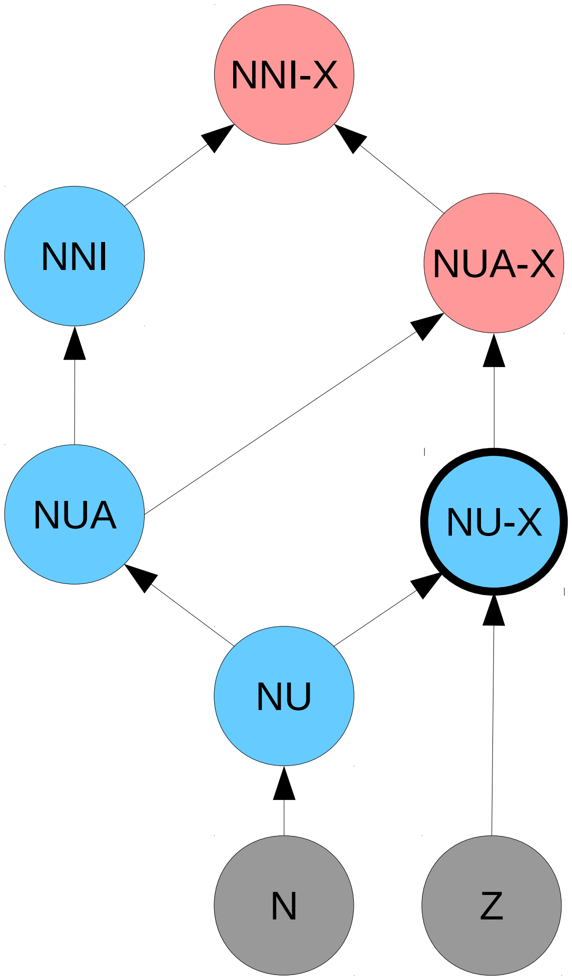
Section 4
Results

Average reduction of the number of nodes in four benchmarks

	GroBdd vs RoBdd
arithmetic	-40,43%
mcnc	-14,74%
iscas99	-25,47%
satlib/uf20-91	-2,86%

Conclusion

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



Extended Results

	Z	NU	NNI	NU-X
arithmetic	44,10%	-40,43%	-74,17%	-55,69%
mcnc	129,66%	-14,74%	-46,86%	-51,79%
iscas99	162,20%	-25,47%	-56,47%	-55,01%
satlib/uf20-91	-41,02%	-2,86%	-29,50%	-93,00%
(log10)	Z	NU	NNI	NU-X
arithmetic	0,15	-0,44	-1,00	-0,60
mcnc	0,09	-0,07	-0,42	-0,35
iscas99	0,36	-0,15	-0,49	-0,40
satlib/uf20-91	-0,24	-0,01	-0,17	-1,19