Chapter 2

Boolean Decision Making by Satisfiability (SAT) Solving

The logician's point of view

Robinson Crusoe

To escape from the island (*a*)

$(a \Rightarrow b \vee c)$ he needs to build a boat (*b*)
or must be discovered by a passing-by ship (*c*)

$(b \Rightarrow d)$ To build a boat (*b*)
he needs wood (*d*)

$(c \Rightarrow e)$ To be discovered by a passing-by ship (*c*)
he needs to send smoke signals (*e*)

$(e \Rightarrow d)$ To send smoke signals (*e*)
he needs wood (*d*)

Robinson's premises: $(a \Rightarrow b \vee c) \wedge (b \Rightarrow d) \wedge (c \Rightarrow e) \wedge (e \Rightarrow d)$

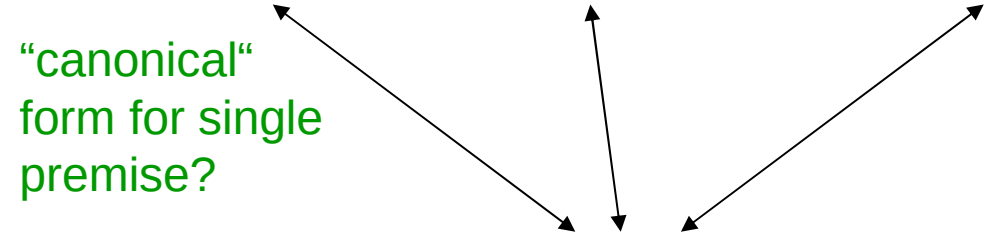
Notation: $\wedge$: "and", $\vee$: "or", $\neg$: "not"

The logician's point of view

Take $(a \Rightarrow b \vee c)$:

the following formulas are equivalent:

$(a \Rightarrow b \vee c)$, $(\neg c \Rightarrow \neg a \vee b)$, $(\neg b \Rightarrow \neg a \vee c)$



"canonical" form for single premise?

Clause! (disjunction of literals)

$(\neg a \vee b \vee c)$

The logician's point of view

Robinson's premises

$$(a \Rightarrow b \vee c) \wedge (b \Rightarrow d) \wedge (c \Rightarrow e) \wedge (e \Rightarrow d)$$

CNF (Conjunctive Normal Form)

$$(\neg a \vee b \vee c) \wedge (\neg b \vee d) \wedge (\neg c \vee e) \wedge (\neg e \vee d)$$

SAT-problem

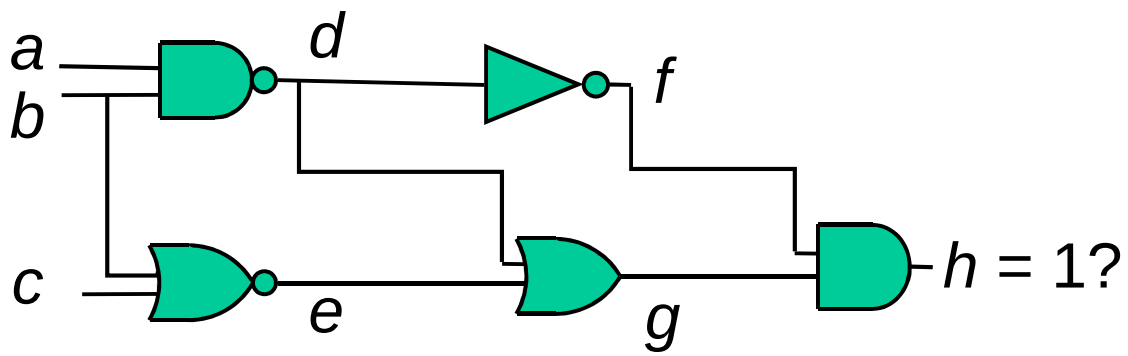
Find a set of assignments for the variables of a formula *f* in CNF such that *f* = 1

or

prove that no such set exists

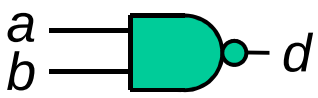
Is the set of premises contradictory?

Circuit satisfiability



$$\varphi = h\,[d = \neg(ab)]\,[e = \neg(b + c)]\,[f = \neg d]\,[g = d + e]\,[h = fg]$$

CNF of a gate



$$\varphi_d = [d = \neg(a\ b)]$$

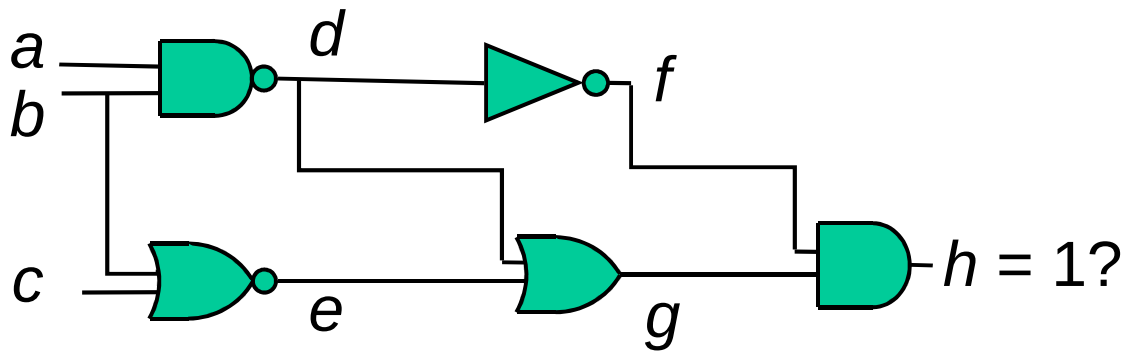
$$= \neg[d \oplus \neg(a\ b)]$$

$$= \neg[\neg(a\ b)\neg d + a\ b\ d]$$

$$= \neg[\neg a\ \neg d + \neg b\ \neg d + a\ b\ d]$$

$$= (a + d)(b + d)(\neg a + \neg b + \neg d)$$

CNF for circuit



$\varphi = h\ [d=\neg(ab)]\ [e=\neg(b+c)]\ [f=\neg d]\ [g=d+e]\ [h=fg]$

$$
\begin{aligned}
= & \ h \\
& (a + d)(b + d)(\neg a + \neg b + \neg d) \\
& (\neg b + \neg e)(\neg c + \neg e)(b + c + e) \\
& (\neg d + \neg f)(d + f) \\
& (\neg d + g)(\neg e + g)(d + e + \neg g) \\
& (f + \neg h)(g + \neg h)(\neg f + \neg g + h)
\end{aligned}
$$

CNF or „clause system“ for circuit and satisfiability of *h*: *“Tseitin construction”*

Classification of Literals and Clauses

A *literal* is a variable or its complement.

A literal is called satisfied, if it has been assigned to value 1.
A literal is called violated, if it has been assigned to value 0.
A literal is called unspecified, if it has not yet been assigned any value.

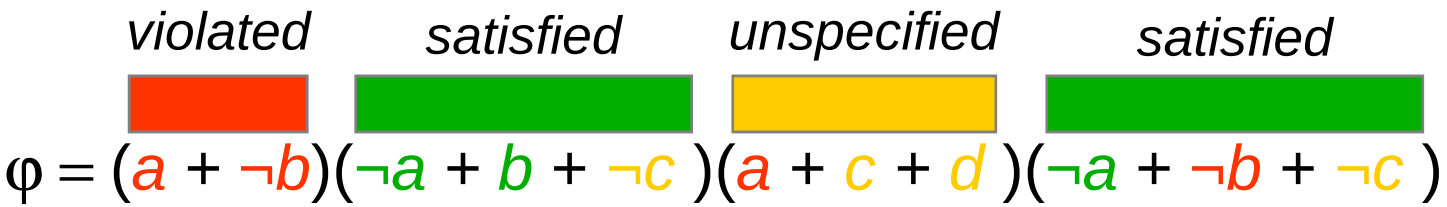
A clause is called satisfied, if it assumed the value 1.
A clause is called violated, if it assumes the value 0.
A clause is called unspecified, if it has not yet assumed a unique logic value.

Example

$\varphi = (a + \neg b)(\neg a + b + \neg c\,)(a + c + d\,)(\neg a + \neg b + \neg c\,)$

Status of clauses and literals after the following value assignment:

$a = 0$
$b = 1$
c: no value assigned yet



SAT solving

Most modern SAT solvers are based on the **Davis-Logemann-Loveland (DLL) Procedure, 1962**

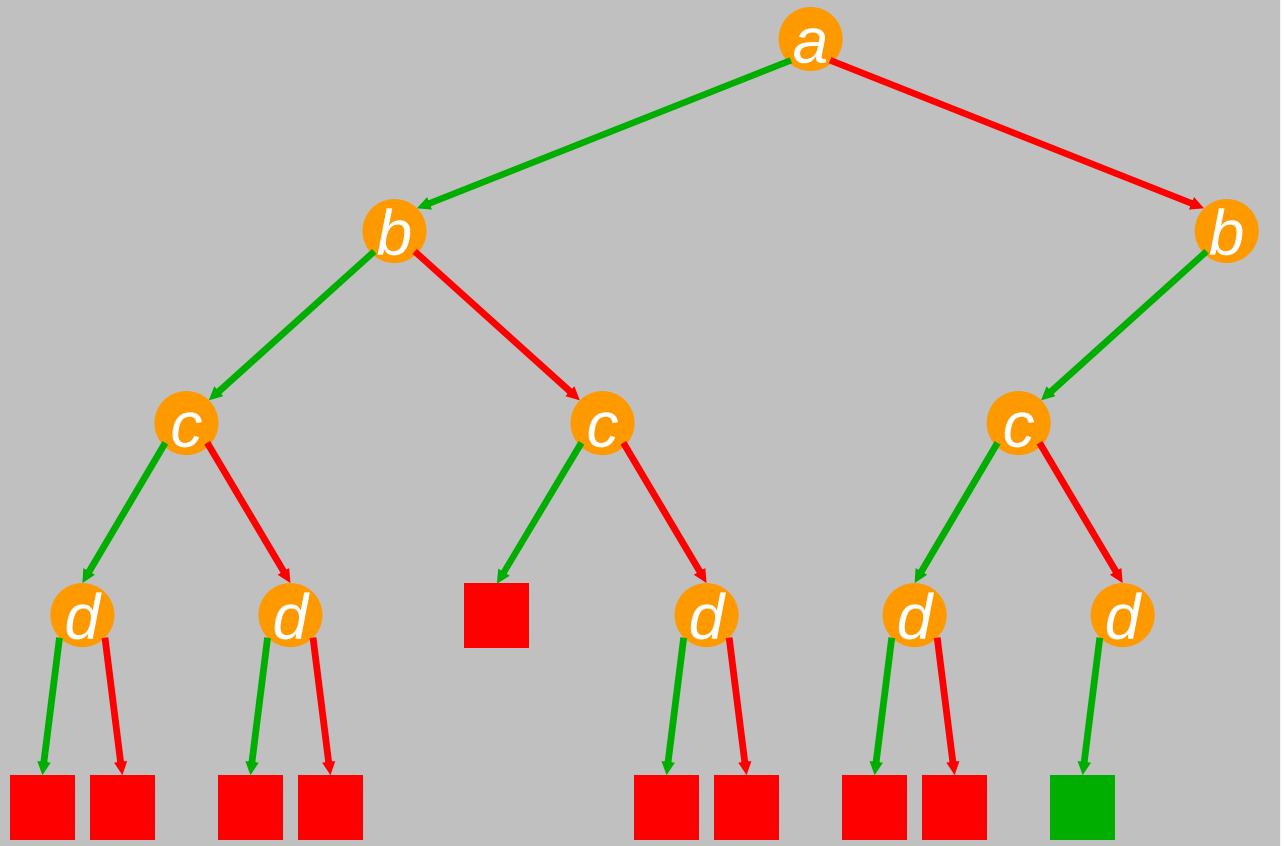
Basic procedure:

Assign step by step values 0 or 1 to the variables in the CNF until all **clauses** are **satisfied** (evaluate to 1).

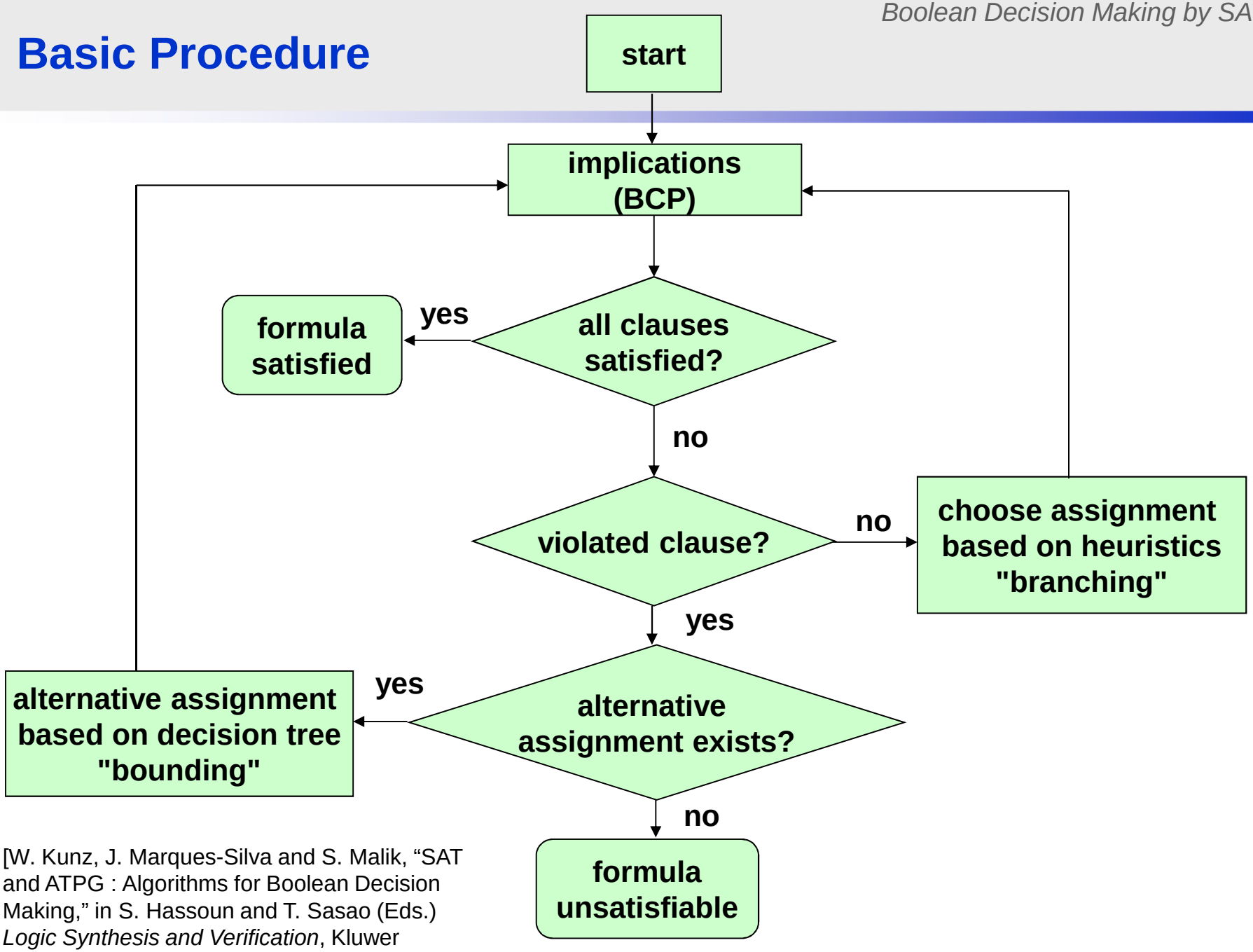
If a **clause** evaluates to 0 it is called **violated** and **backtracks** are performed based on a **decision tree**.

Backtracks in decision tree

1. $(a + b + c)$
2. $(a + b + \neg c)$
3. $(\neg a + b + \neg c)$
4. $(a + c + d)$
5. $(\neg a + c + d)$
6. $(\neg a + c + \neg d)$
7. $(\neg b + \neg c + \neg d)$
8. $(\neg b + \neg c + d)$

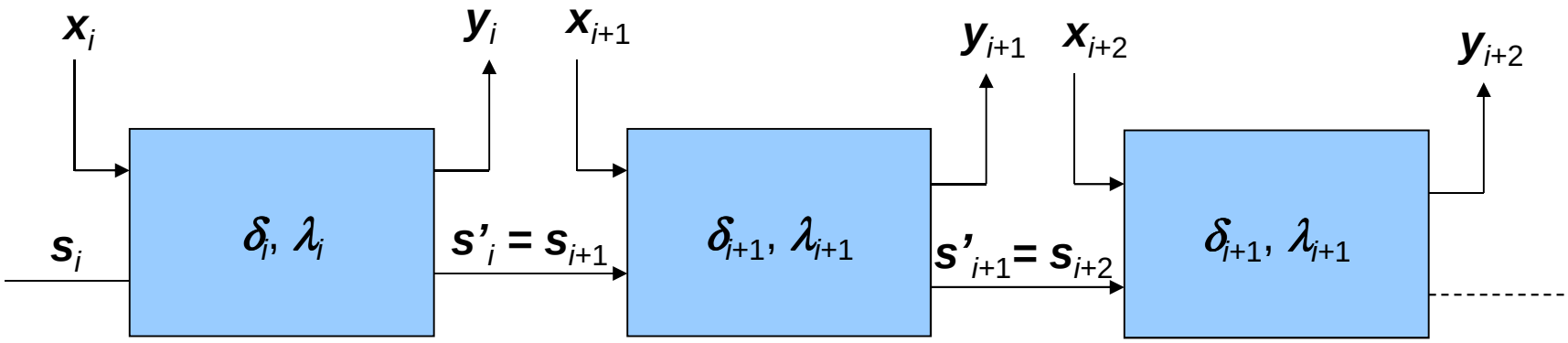


Basic Procedure



[W. Kunz, J. Marques-Silva and S. Malik, “SAT and ATPG : Algorithms for Boolean Decision Making,” in S. Hassoun and T. Sasao (Eds.) *Logic Synthesis and Verification*, Kluwer Academic Publishers, 2002.]

SAT-based property checking: the model

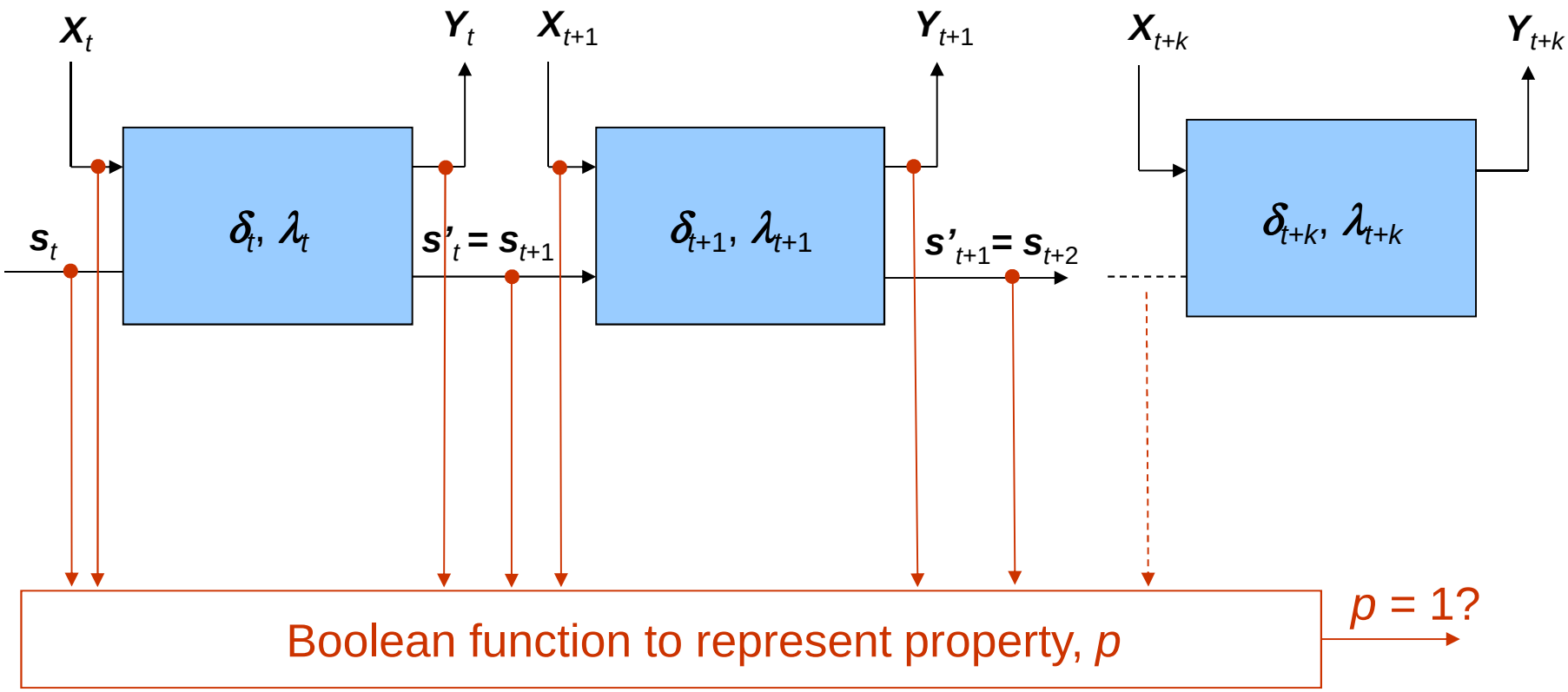


iterative circuit array

Concatenate *k* copies of the combinational logic for δ and λ („time frames“)
$\Rightarrow$ no feedback loops, entirely combinational model

"Interval Property Checking (IPC)"

iterative circuit array from $i = t$ to $i = t + k$



Compute CNF for iterative model and property