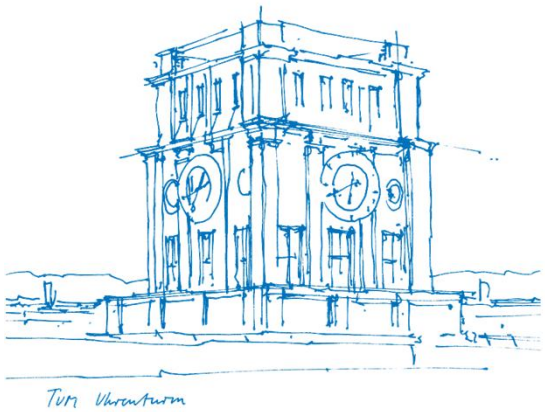
Limit-Deterministic Büchi Automata for Probabilistic Model Checking

Javier Esparza Jan Křetínský

Stefan Jaax Salomon Sickert

Technische Universität München



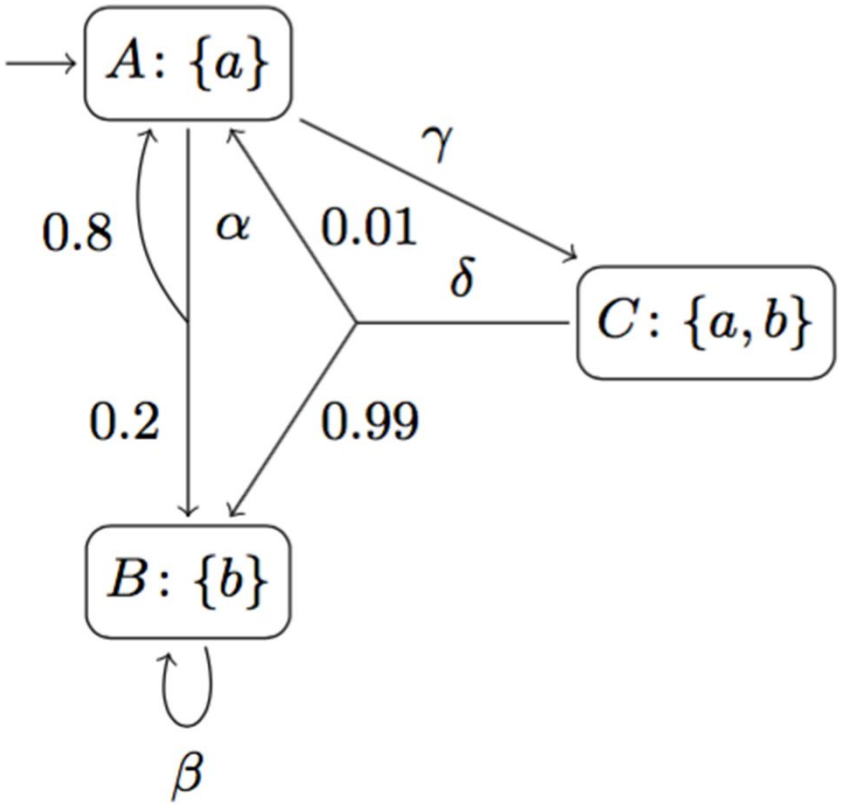
PROBABILISTIC MODEL CHECKING

- Markov Decision Process (MDP) .

  At each state, a scheduler chooses a probability distribution, and then the next state is chosen stochastically according to the distribution.

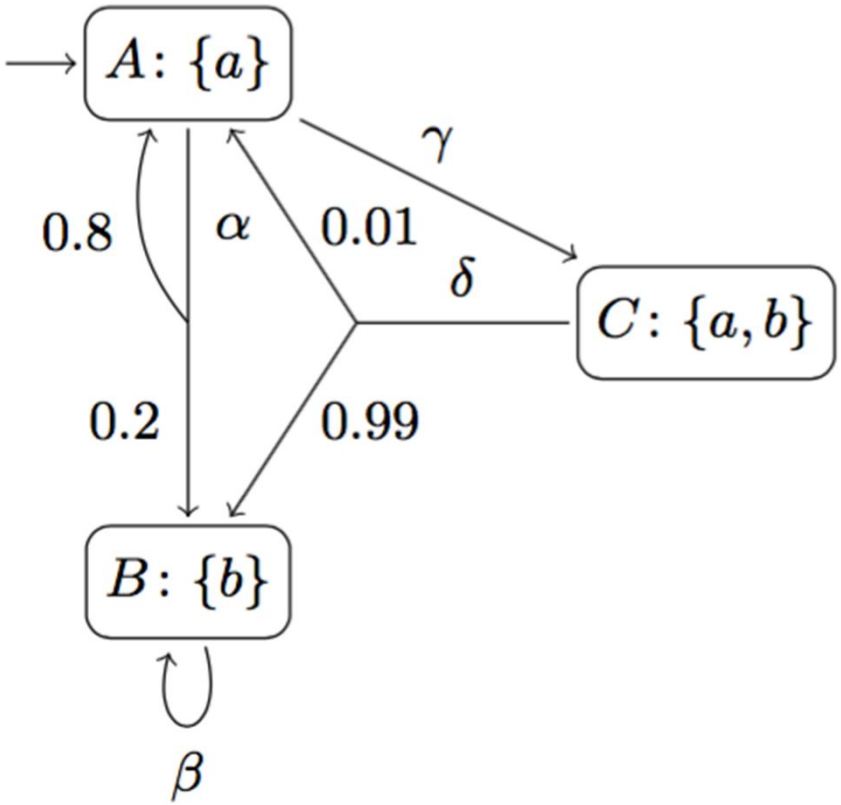
- For a fixed scheduler:

  MDP → Markov chain



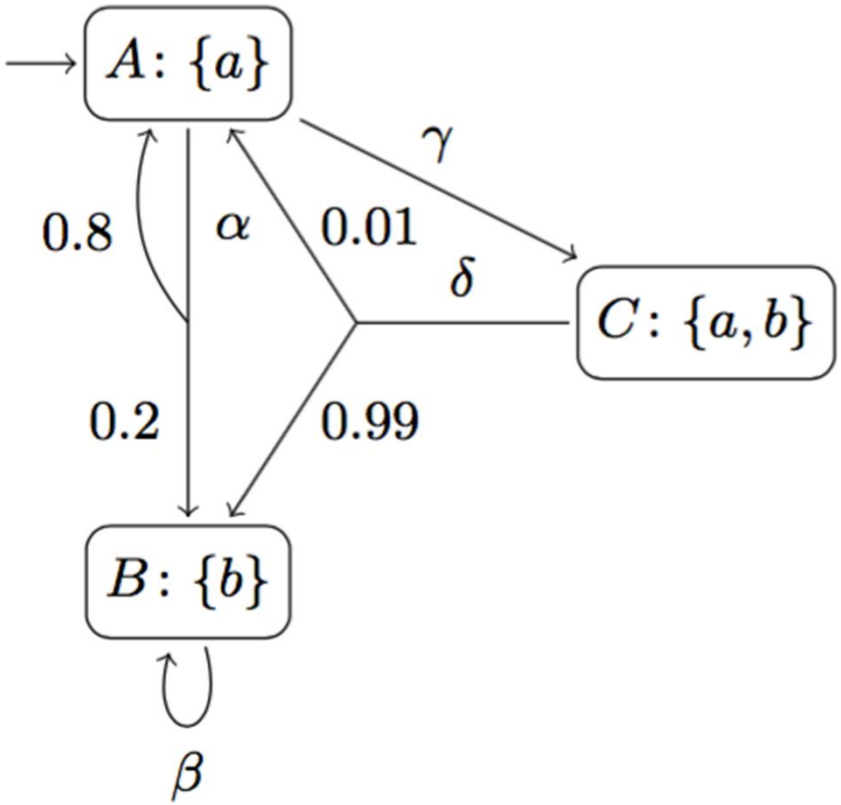
PROBABILISTIC MODEL CHECKING

- Qualitative Model Checking:
  - Input: MDP, LTL formula
  - Does the formula hold for all schedulers with probability 1?

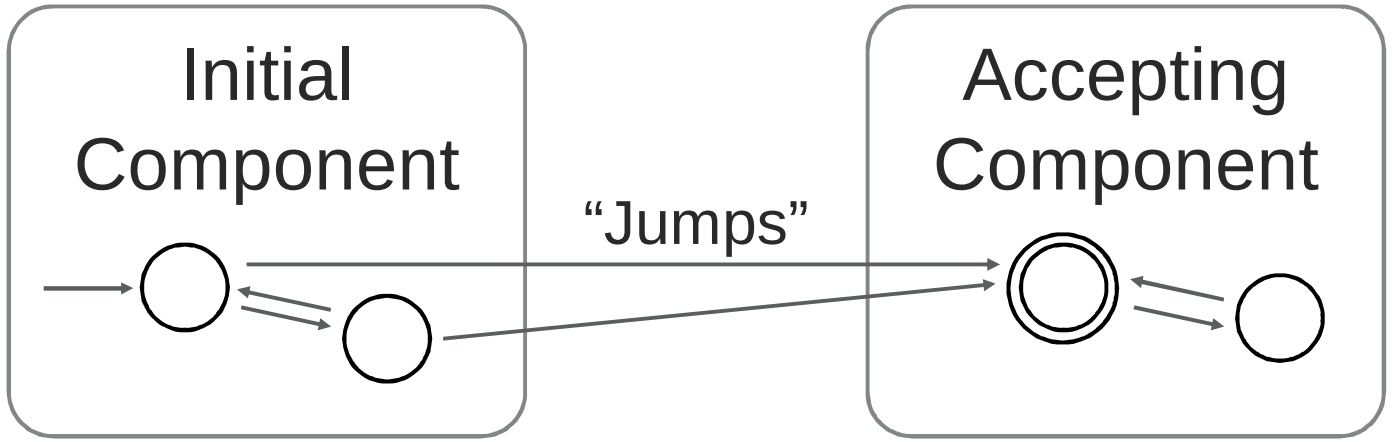


PROBABILISTIC MODEL CHECKING

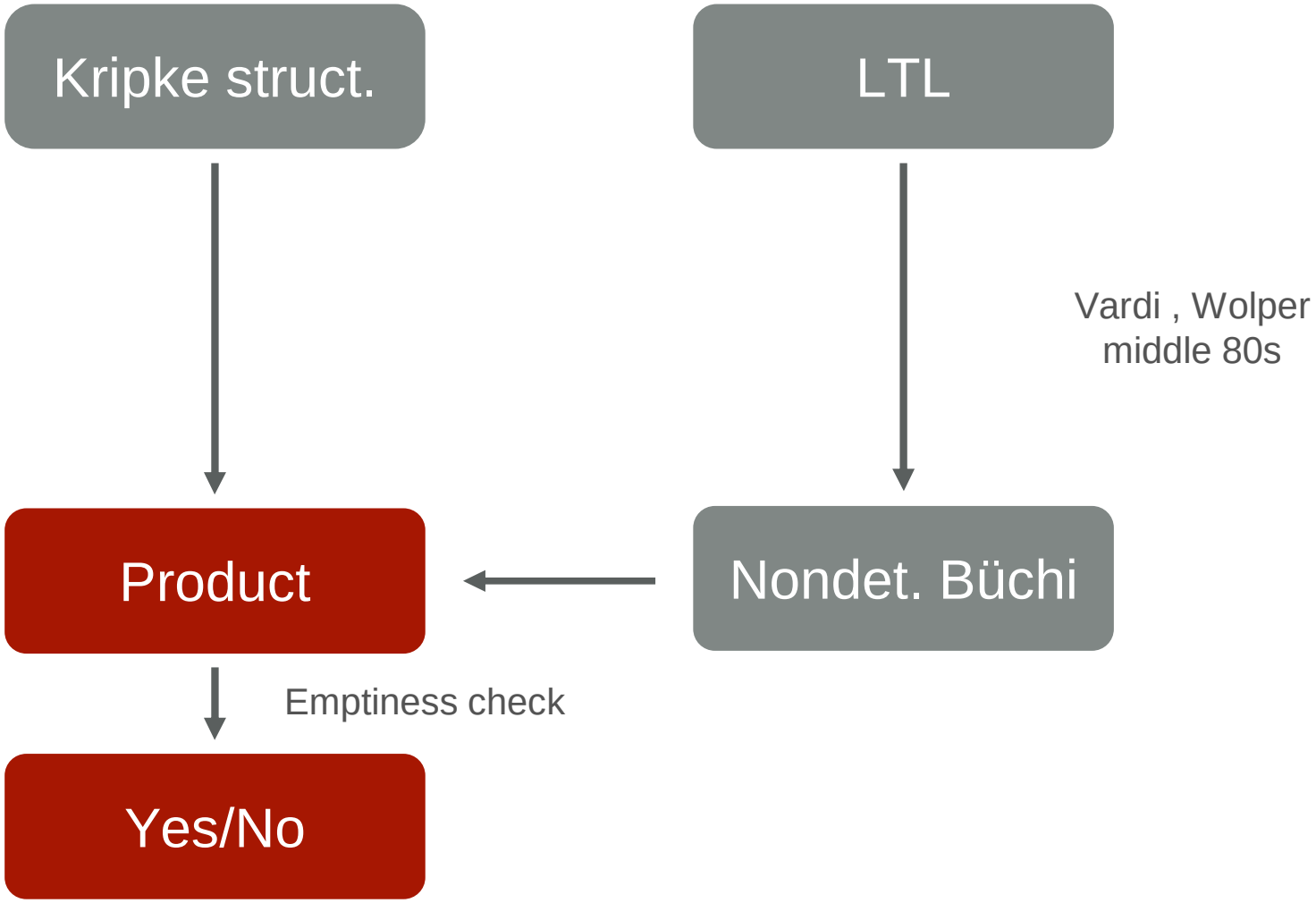
- Qualitative Model Checking:
  - Input: MDP, LTL formula
  - Does the formula hold for all schedulers with probability 1?
- Quantitative Model Checking:
  - Input: MDP, LTL formula, threshold c
  - Does the formula hold for all schedulers with probability at least c?



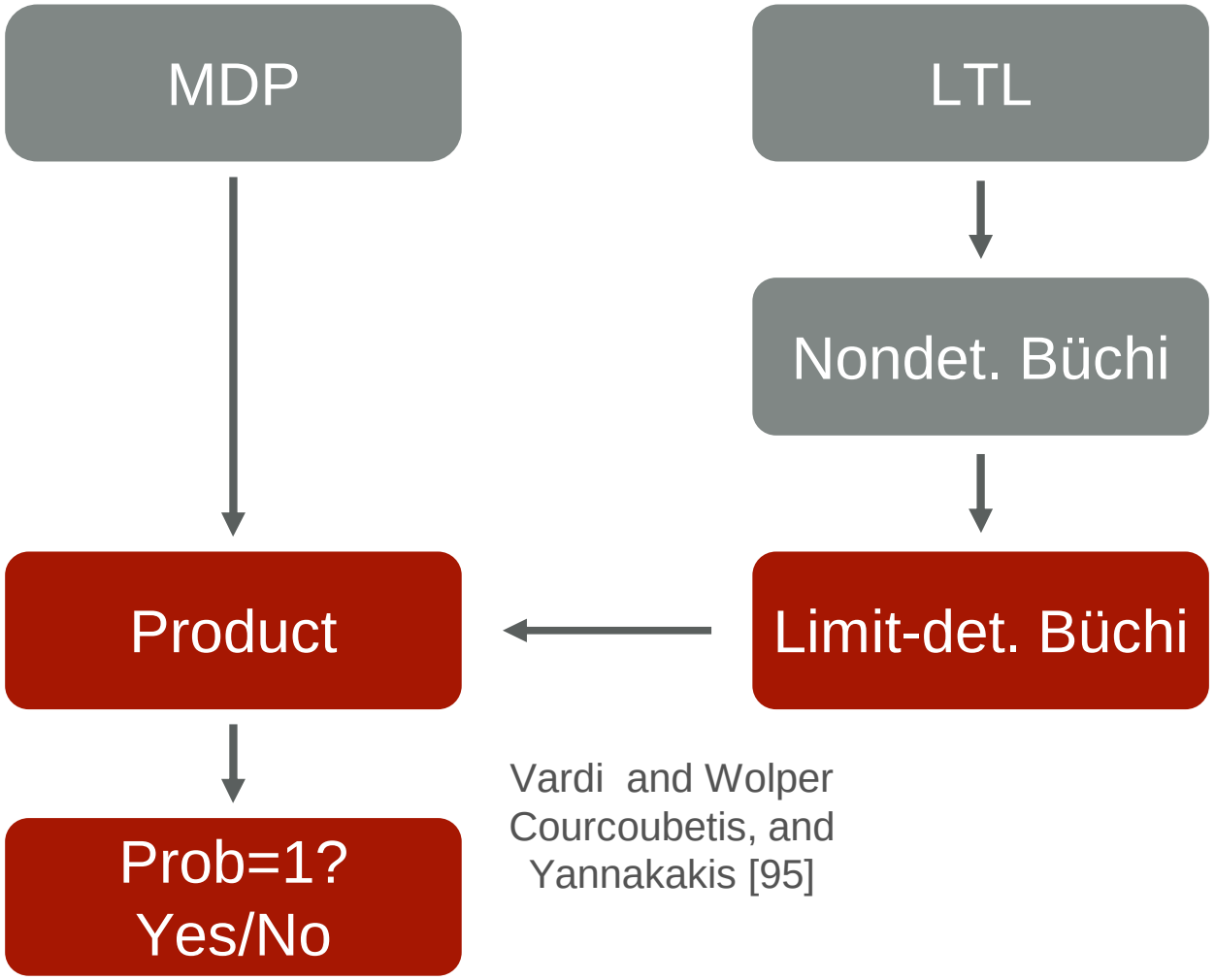
LIMIT-DETERMINISTIC BÜCHI AUTOMATA



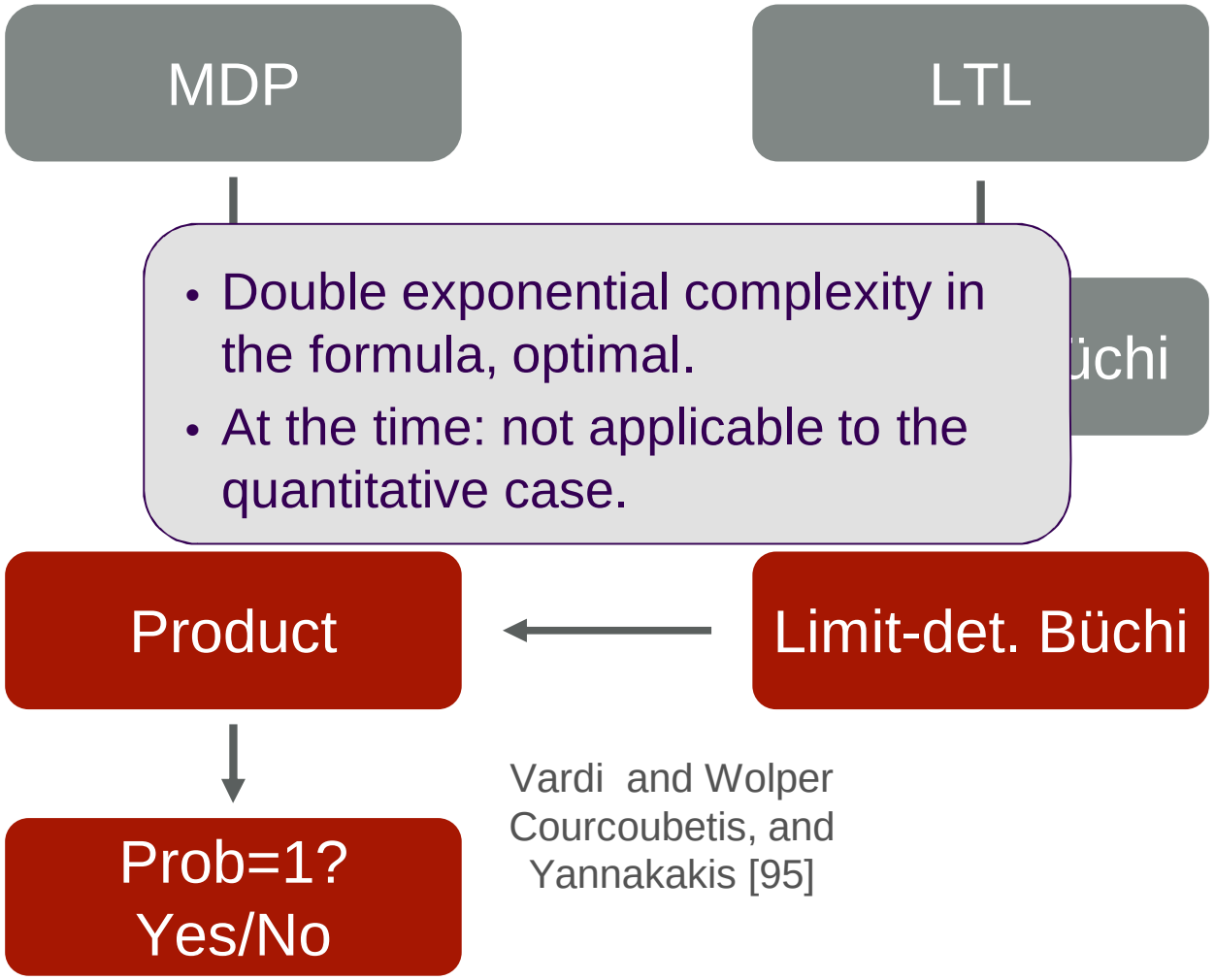
AUTOMATA-BASED MODEL CHECKING



QUALITATIVE
PROB. MODEL CHECKING



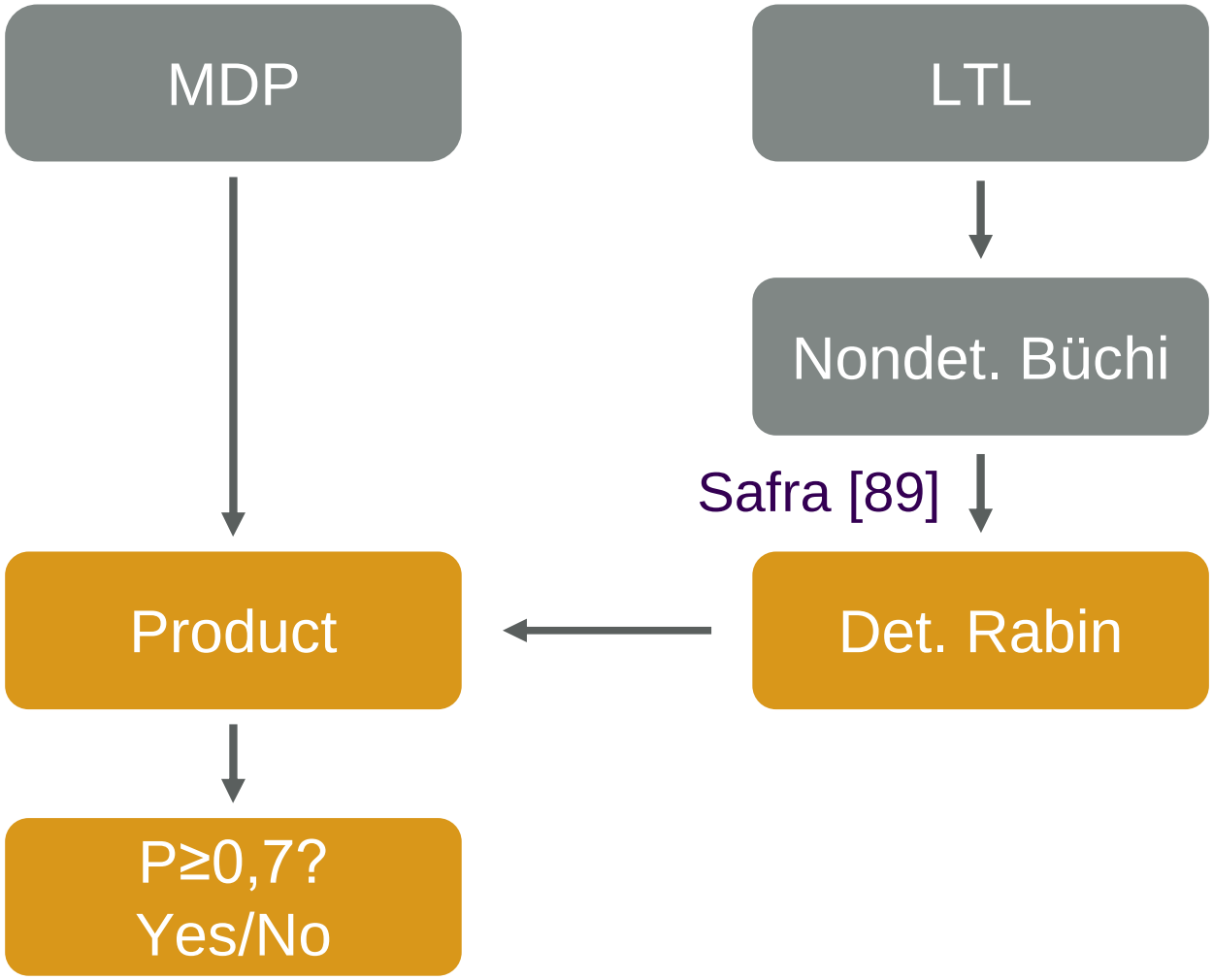
QUALITATIVE
PROB. MODEL CHECKING



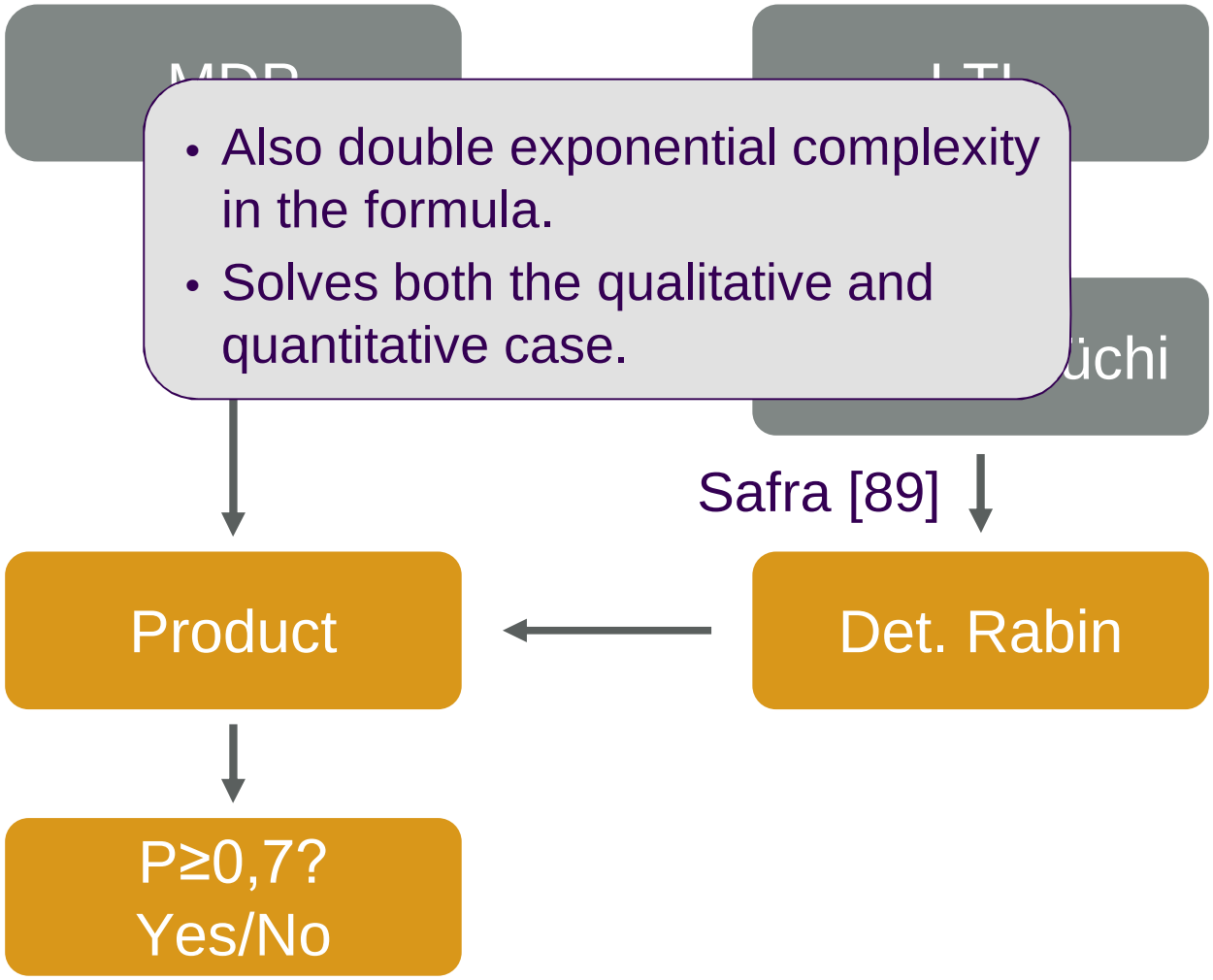
Vardi and Wolper
Courcoubetis, and
Yannakakis [95]

Vardi and Wolper
Courcoubetis, and
Yannakakis [95]

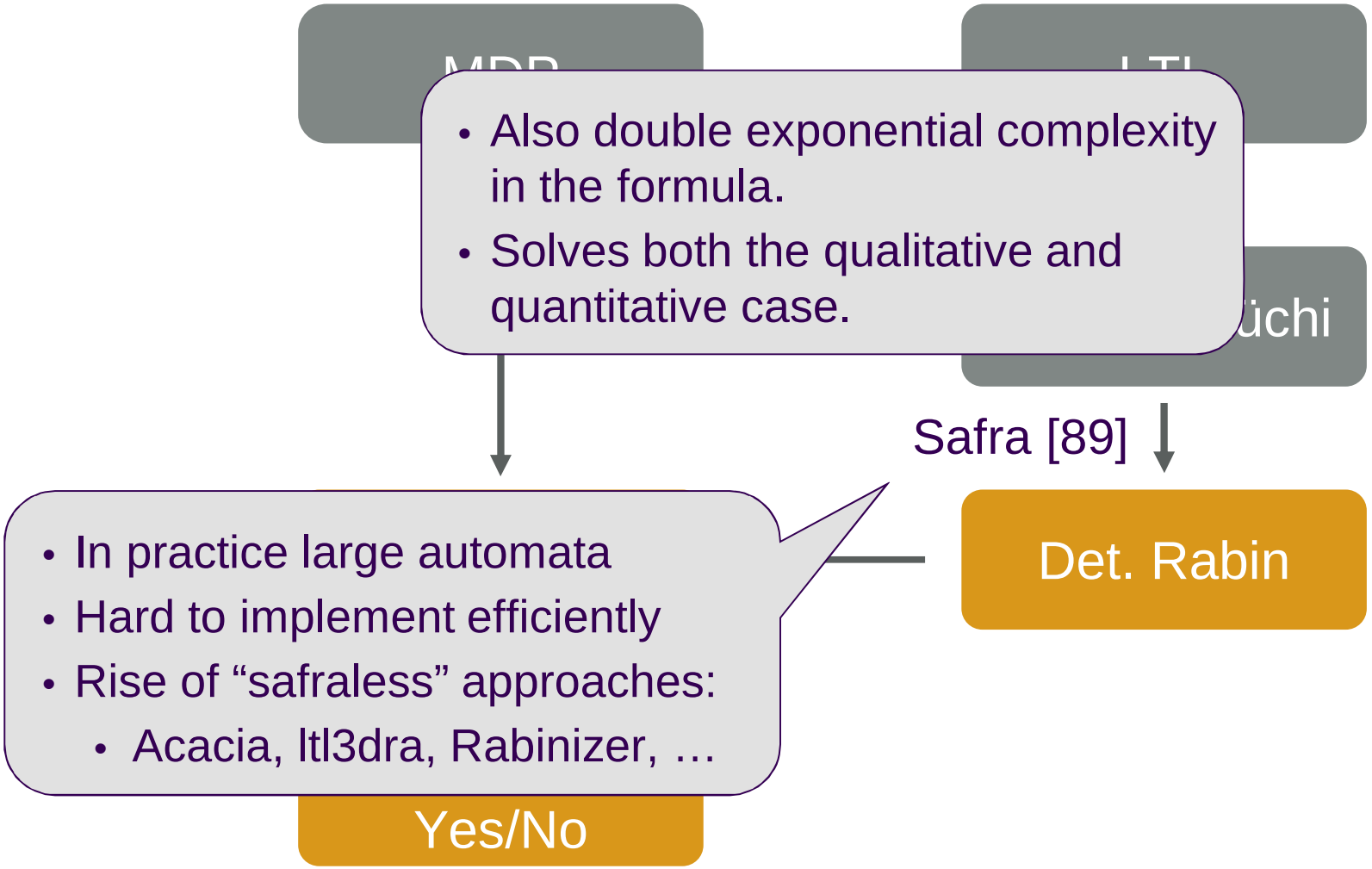
QUANTITATIVE
PROB. MODEL CHECKING



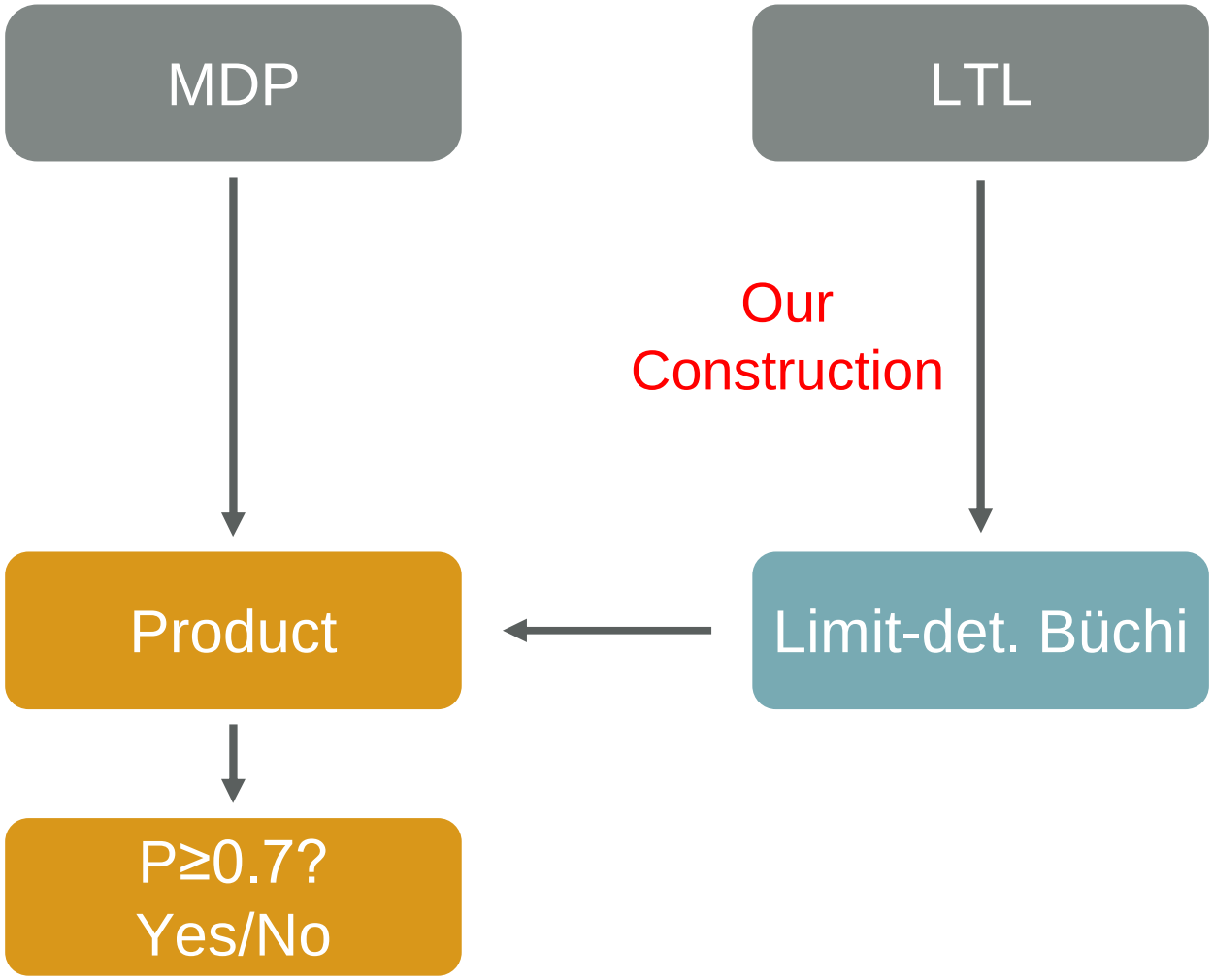
QUANTITATIVE
PROB. MODEL CHECKING



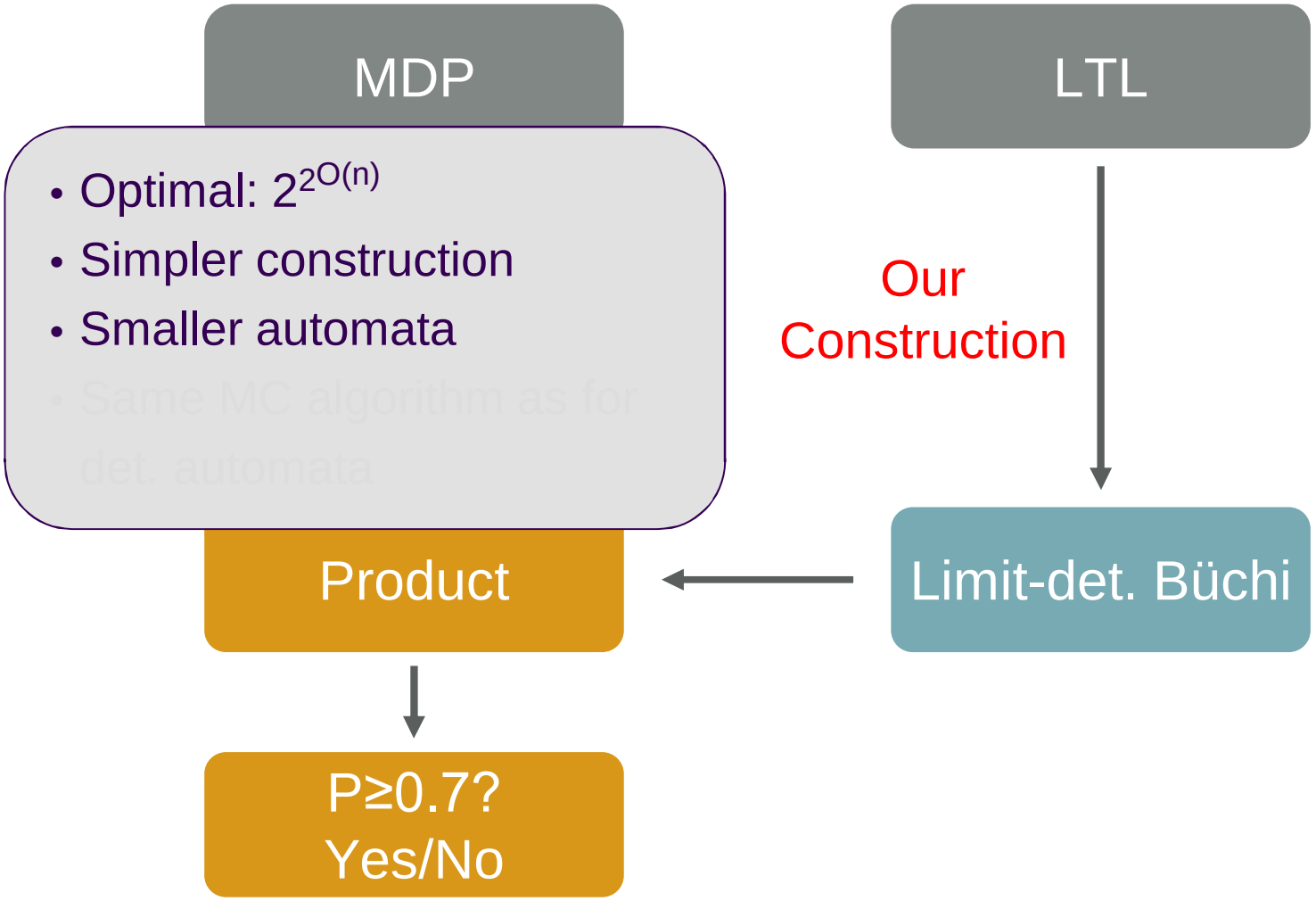
QUANTITATIVE
PROB. MODEL CHECKING



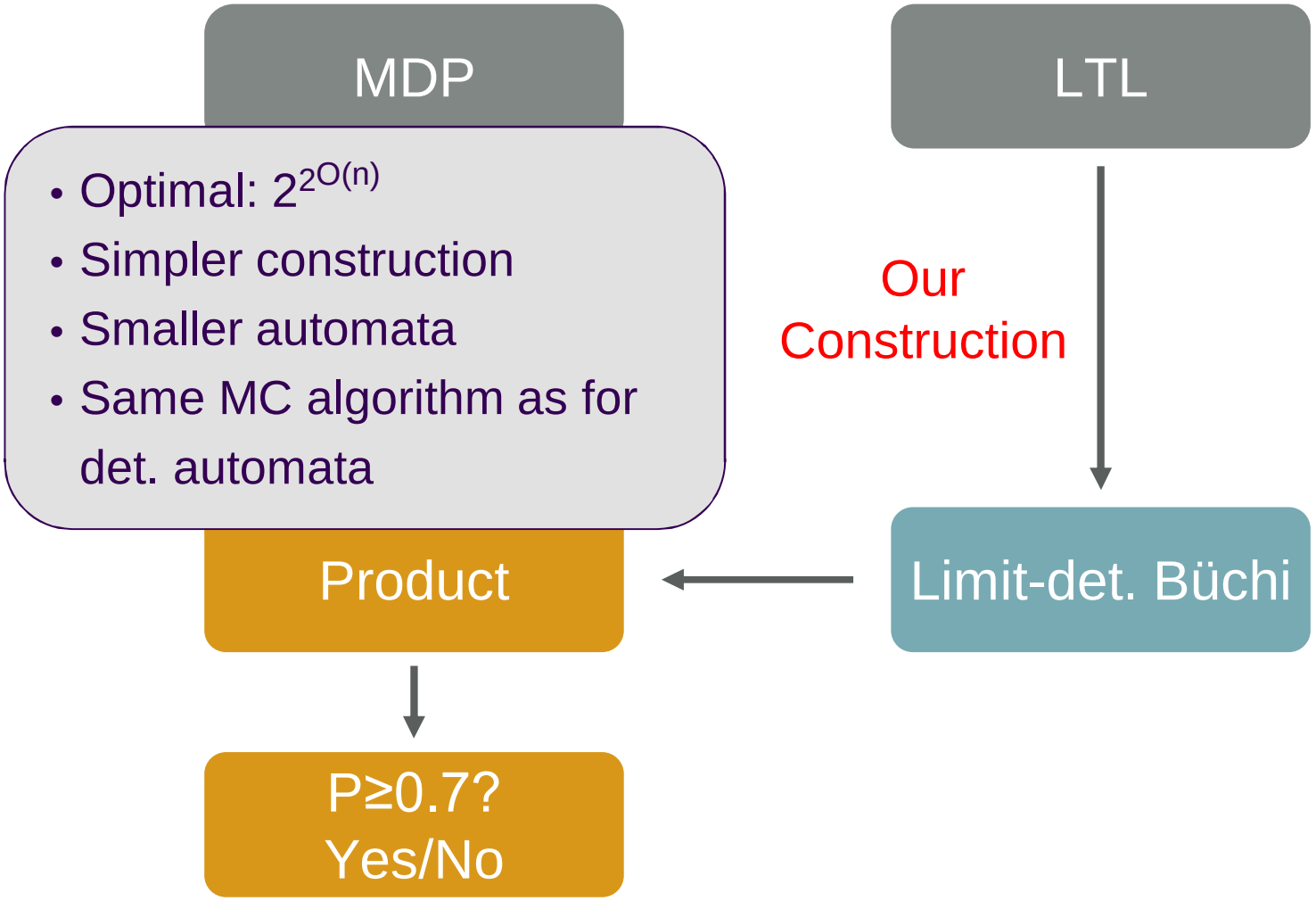
QUANTITATIVE PROB. MODEL CHECKING



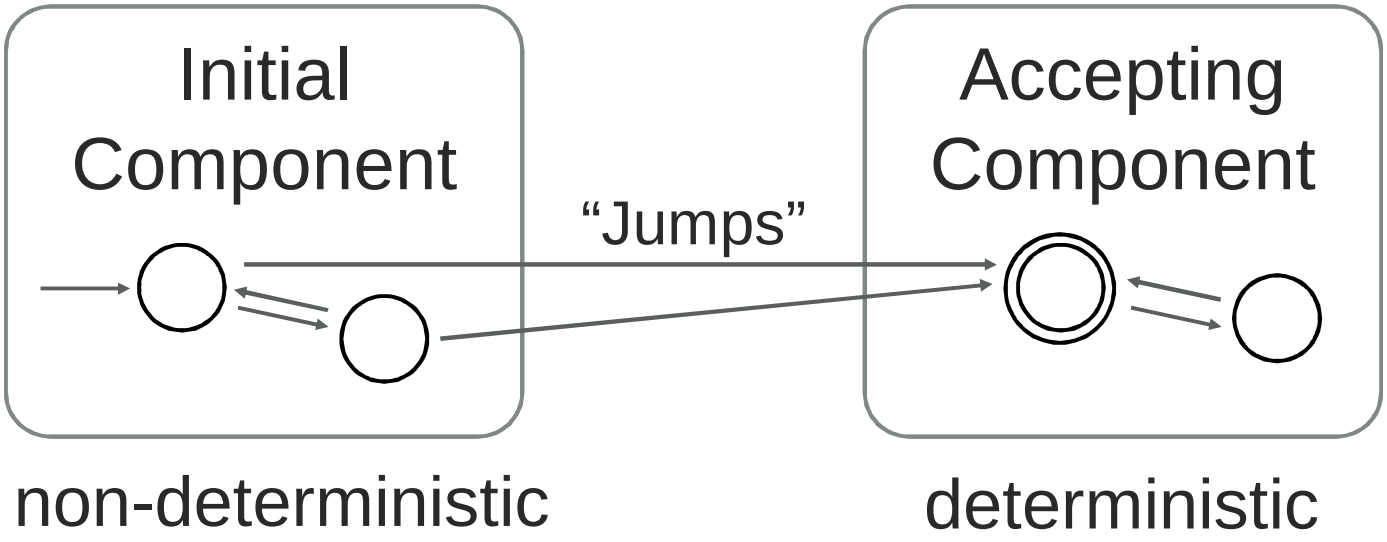
QUANTITATIVE PROB. MODEL CHECKING



QUANTITATIVE PROB. MODEL CHECKING

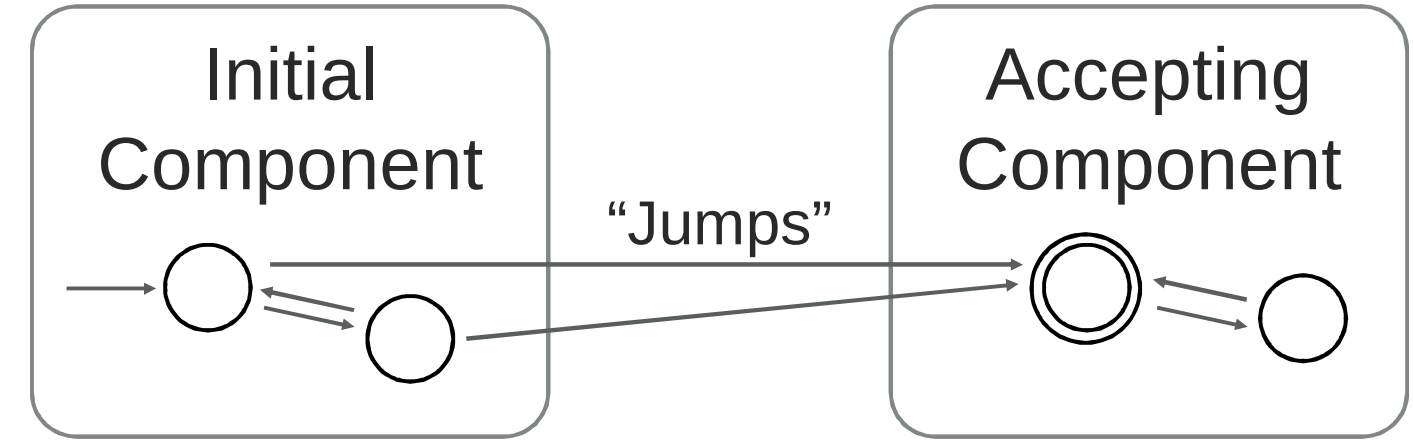


LIMIT-DETERMINISM



LIMIT-DETERMINISM

In our construction:



~~non-deterministic~~

deterministic

deterministic

Every runs „uses“ nondeterminism at most once

PRELIMINARIES

- Linear Temporal Logic in Negation Normal Form

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

PRELIMINARIES

- Linear Temporal Logic in Negation Normal Form

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

- Monotonicity of NNF:

  if w satisfies φ

  w' satisfies all the subformulas of φ satisfied by w, and perhaps more

  then w' satisfies φ

PRELIMINARIES

- Linear Temporal Logic in Negation Normal Form

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

Only liveness operator.

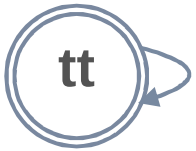
- Monotonicity of NNF:

  if w satisfies φ

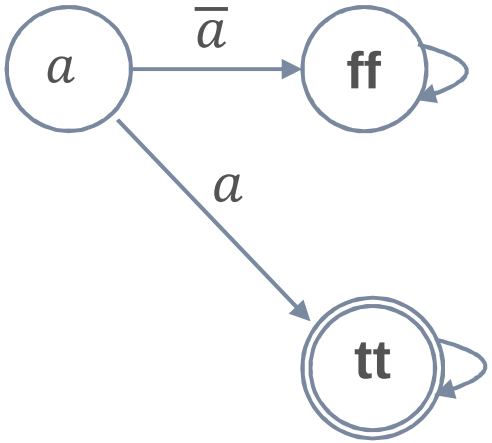
  w' satisfies all the subformulas of φ satisfied by w, and perhaps more

  then w' satisfies φ

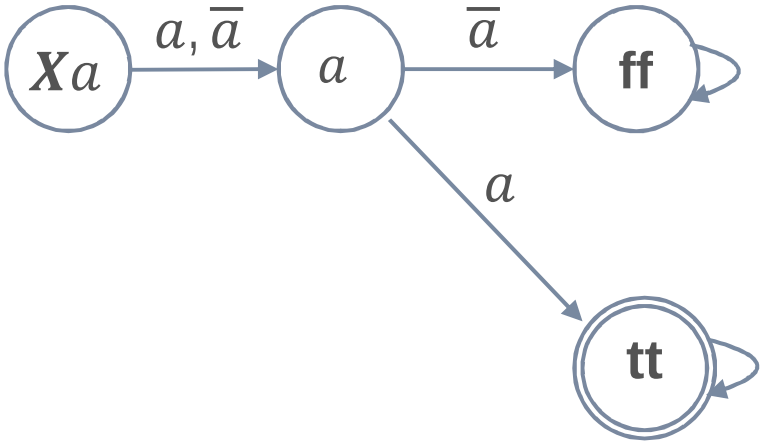
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



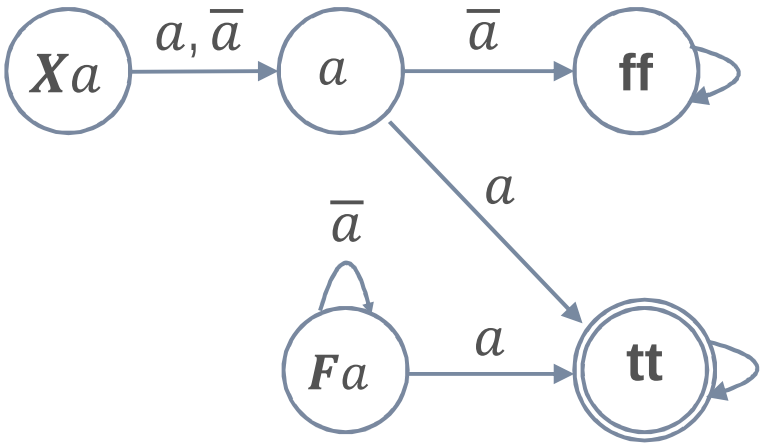
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



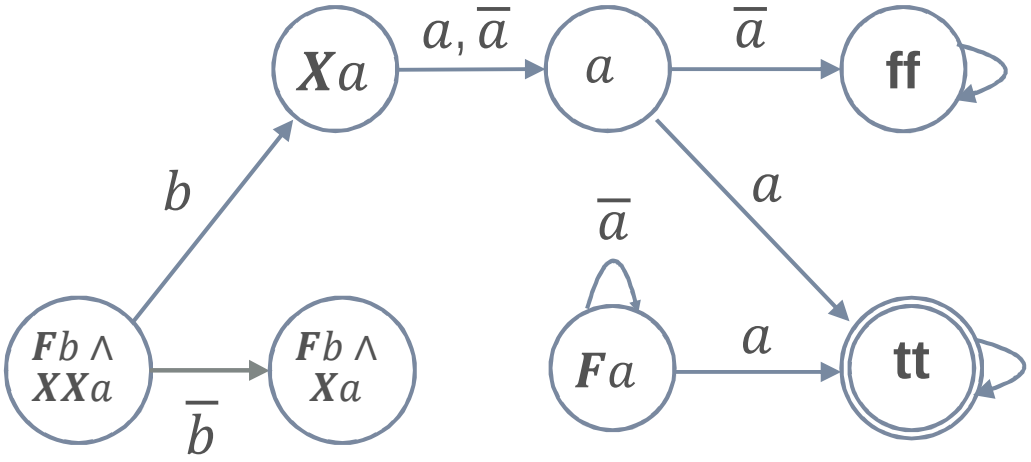
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON

The formula $af(\varphi, \nu)$ (“φ after ν”) is defined by:

$$af(\mathbf{tt}, \nu) = \mathbf{tt} \qquad af(\mathbf{ff}, \nu) = \mathbf{ff}$$

$$af(a, \nu) = \begin{cases} \mathbf{tt} & if\ a \in \nu \\ \mathbf{ff} & if\ a \notin \nu \end{cases} \qquad af(\neg a, \nu) = \begin{cases} \mathbf{ff} & if\ a \in \nu \\ \mathbf{tt} & if\ a \notin \nu \end{cases}$$

$$af(\varphi \wedge \psi, \nu) = af(\varphi, \nu) \wedge af(\psi, \nu)$$

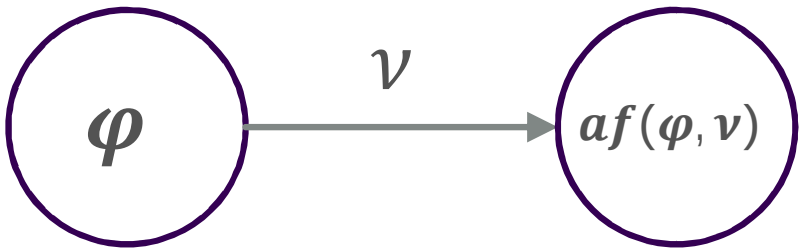
$$af(\varphi \vee \psi, \nu) = af(\varphi, \nu) \vee af(\psi, \nu)$$

$$af(\mathbf{X}\varphi, \nu) = \varphi$$

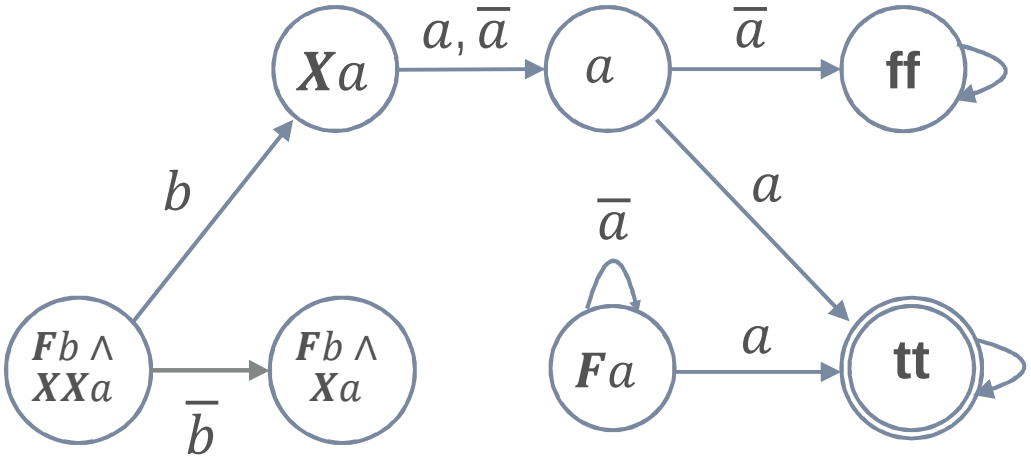
$$af(\mathbf{G}\varphi, \nu) = af(\varphi, \nu) \wedge \mathbf{G}\varphi$$

$$af(\mathbf{F}\varphi, \nu) = af(\varphi, \nu) \vee \mathbf{F}\varphi$$

$$af(\varphi \mathbf{U} \psi, \nu) = af(\psi, \nu) \vee (af(\varphi, \nu) \wedge \varphi \mathbf{U} \psi)$$

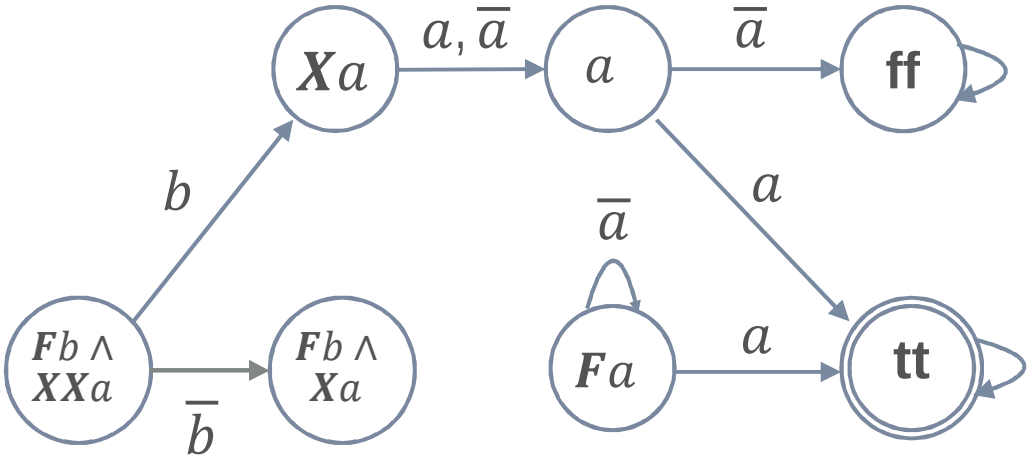


FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



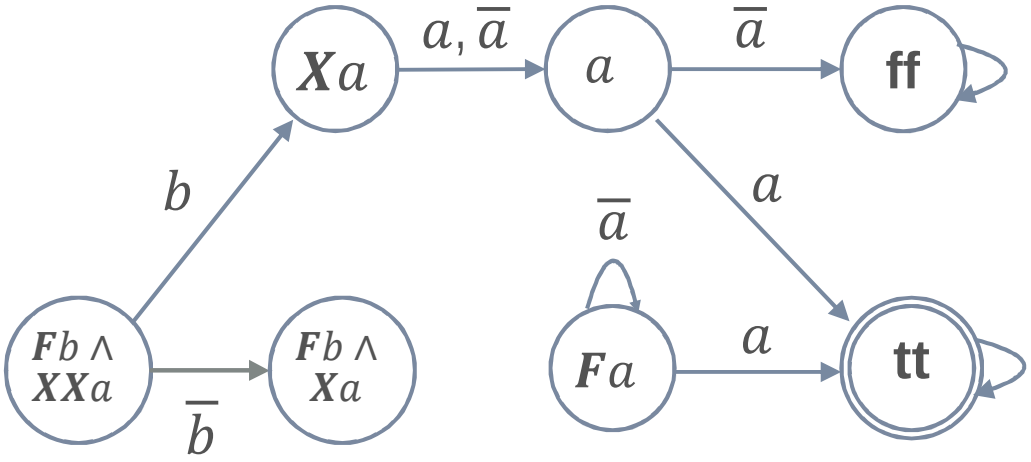
- The automaton „tracks“ the property that must hold now for the original property to hold at the beginning

FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



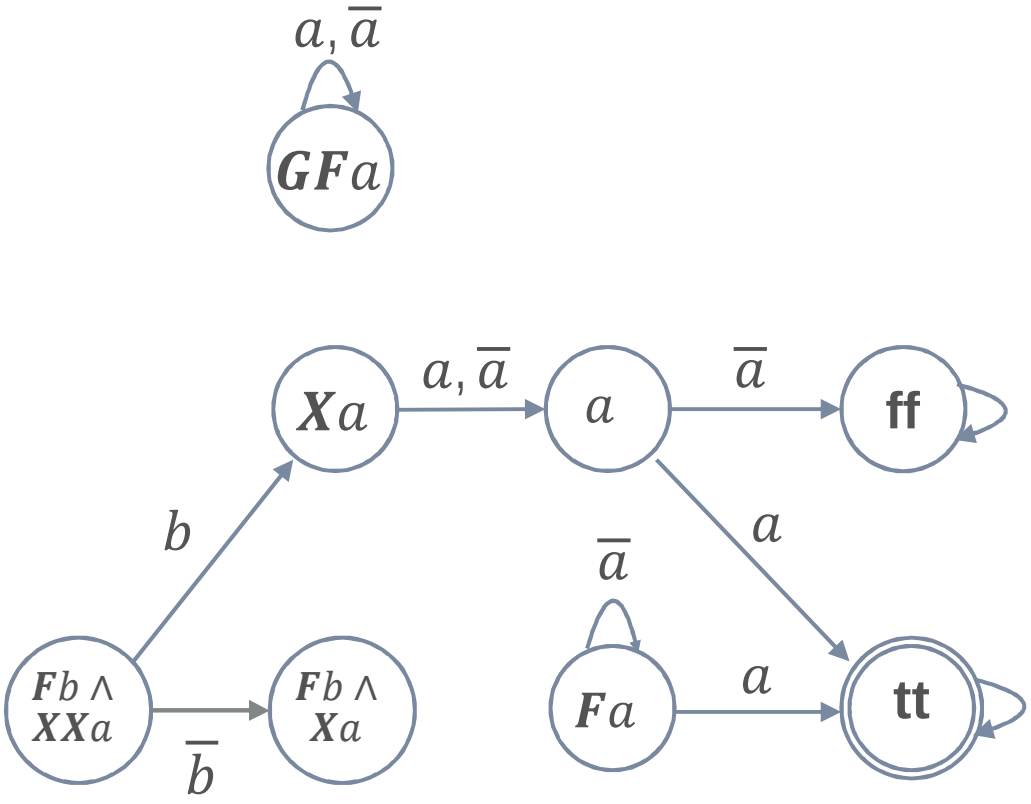
- The automaton „tracks“ the property that must hold now for the original property to hold at the beginning
- Formulas with F, X, U: ✔

FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



- The automaton „tracks“ the property that must hold now for the original property to hold at the beginning
- Formulas with F, X, U: ✔
- Formulas with G: not good enough.

FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



- The automaton „tracks“ the property that must hold now for the original property to hold at the beginning
- Formulas with $\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{U}$: ✔
- Formulas with $\boldsymbol{G}$: not good enough.

$\boldsymbol{G}$-SUBFORMULAS

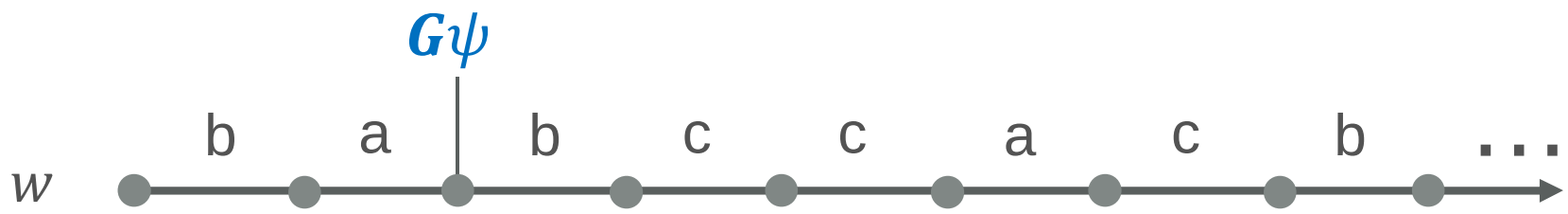
- Fix a formula φ and a word w.
  Let $\boldsymbol{G}\psi$ be a $\boldsymbol{G}$-subformula of φ.

$\boldsymbol{G}$-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $\boldsymbol{G}\psi$ be a $\boldsymbol{G}$-subformula of φ.

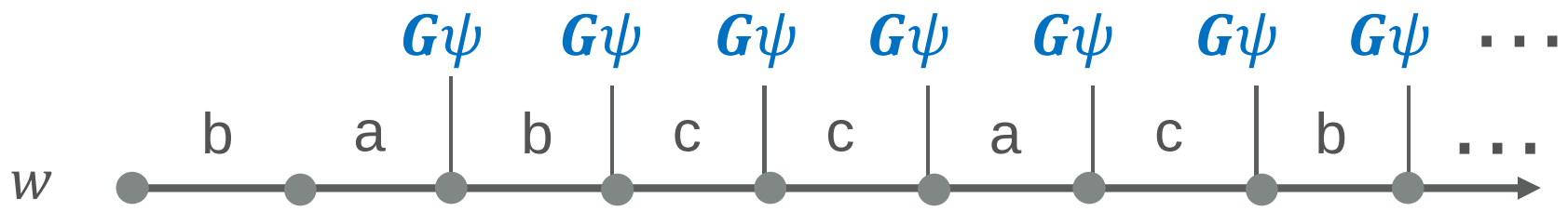
G-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $G\psi$ be a G-subformula of φ.



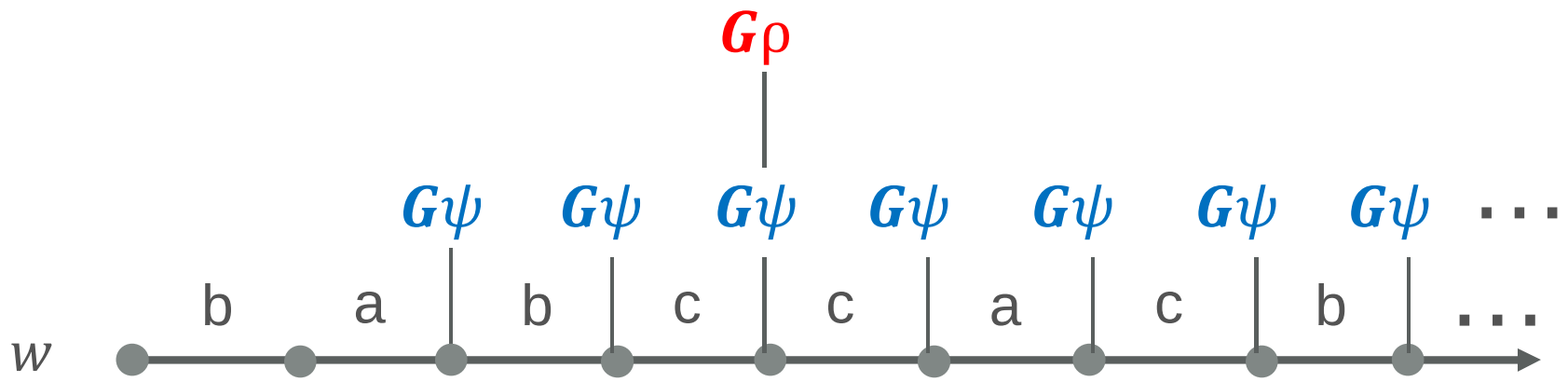
G-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $G\psi$ be a G-subformula of φ.



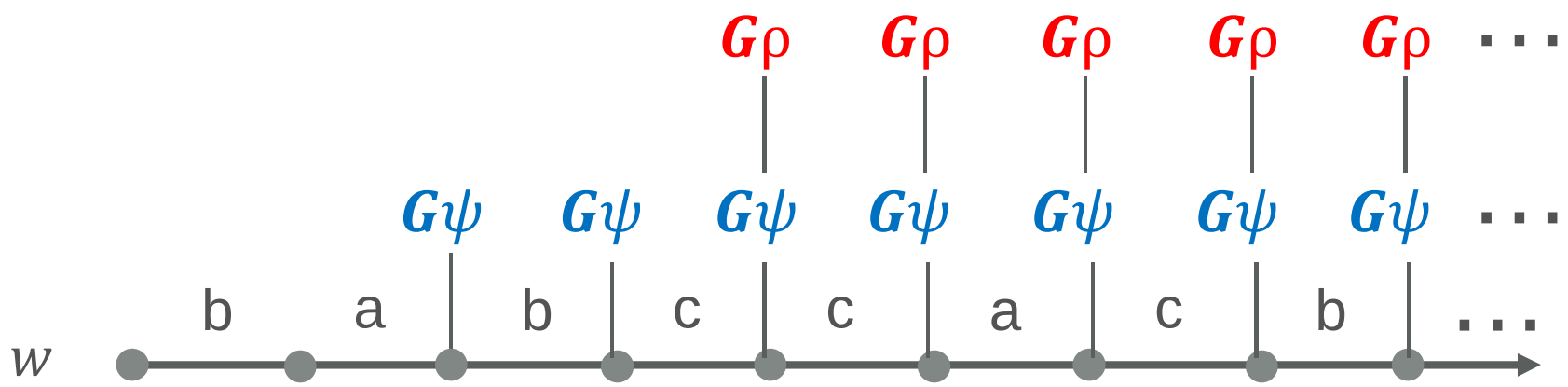
G-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $G\psi$ be a G-subformula of φ.



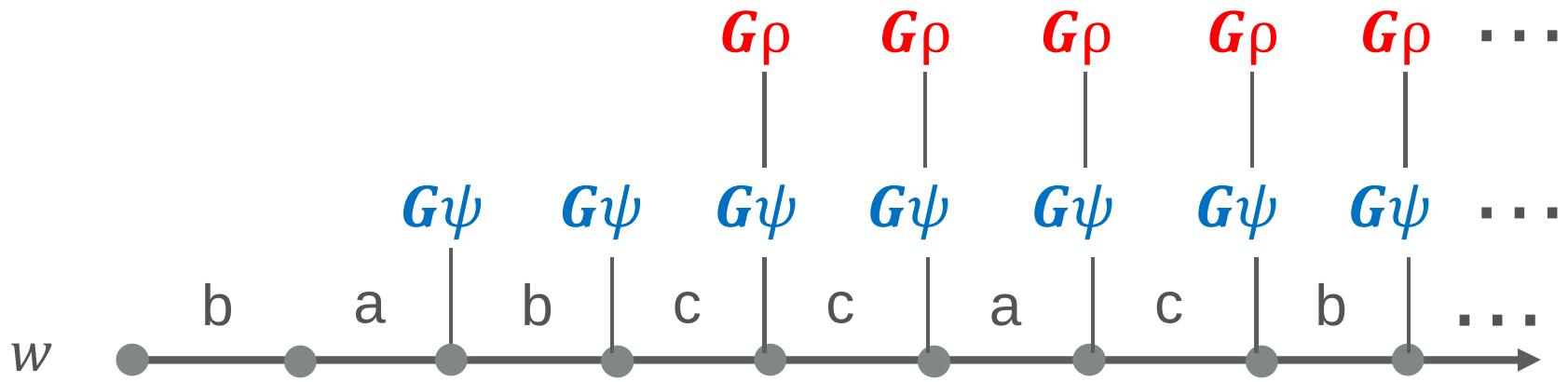
G-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $G\psi$ be a G-subformula of φ.



G-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $G\psi$ be a G-subformula of φ.



- Informally: while reading the word w, the set of G-subformulas that hold cannot decrease, and eventually stabilizes to a set $\mathrm{True}Gs(w,\varphi)$.

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of $\boldsymbol{G}$-subformulas of ψ.

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of $\boldsymbol{G}$-subformulas of ψ.
- „Meaning“ of a $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the word satisfies

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of $\boldsymbol{G}$-subformulas of ψ.
- „Meaning“ of a $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the word satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of $\boldsymbol{G}$-subformulas of ψ.
- „Meaning“ of a $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the word satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of G-subformulas of ψ.
- „Meaning“ of a $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the word satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of G-subformulas of ψ.
- „Meaning“ of a $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the word satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.
- After the jump, the task of the accepting component is to „check that the guess is correct“, i.e., accept iff the guess is correct.

SECOND STEP: JUMPING

- „Meaning“ of the $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the run satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.

SECOND STEP: JUMPING

- „Meaning“ of the $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the run satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.

- $w \vDash \varphi$ iff the automaton can guess correctly.

SECOND STEP: JUMPING

- „Meaning“ of the $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the run satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.

- $w \vDash \varphi$ iff the automaton can guess correctly.
  - If the correct guess is made at suffix w' then $w' \vDash \psi$ which implies $w \vDash \varphi$ (tracking!)

SECOND STEP: JUMPING

- „Meaning“ of the $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the run satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.

- $w \vDash \varphi$ iff the automaton can guess correctly.
  - If the correct guess is made at suffix w' then $w' \vDash \psi$ which implies $w \vDash \varphi$ (tracking!)
  - If $w \vDash \varphi$ then $w' \vDash \mathrm{True}Gs(w, \varphi)$ for some suffix w' and so the jump before w' that chooses $\mathcal{G} \coloneqq \mathrm{True}Gs(w, \varphi)$ satisfies 1. and 2.

A DBA THAT CHECKS 1. & 2.

- Since DBAs are closed under intersection, it suffices to construct two DBAs for 1. and 2.

CHECKING 2.

- „$\mathcal{G} \Rightarrow \psi$ holds even if no other G-subformula of ψ ever becomes true”

CHECKING 2.

- „$\mathcal{G} \Rightarrow \psi$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”
- Example: $\psi = \boldsymbol{G}(a \vee \boldsymbol{F}b) \wedge (\boldsymbol{G}c \vee \boldsymbol{X}d)$

  $\mathcal{G} = \{ \boldsymbol{G}(a \vee \boldsymbol{F}b) \}$

  Reduces to checking $\boldsymbol{X}d$

CHECKING 2.

- „$\mathcal{G} \Rightarrow \psi$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”
- Example: $\psi = \boldsymbol{G}(a \vee \boldsymbol{F}b) \wedge (\boldsymbol{G}c \vee \boldsymbol{X}d)$

  $\mathcal{G} = \{ \boldsymbol{G}(a \vee \boldsymbol{F}b) \}$

  Reduces to checking $\boldsymbol{X}d$
- Reduces to checking the $\boldsymbol{G}$-free formula

  $\psi[\ \mathcal{G} \setminus \mathrm{tt}\ ,\ \overline{\mathcal{G}} \setminus \mathrm{ff}\ ]$

CHECKING 2.

- „$\mathcal{G} \Rightarrow \psi$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”
- Example: $\psi = \boldsymbol{G}(a \vee \boldsymbol{F}b) \wedge (\boldsymbol{G}c \vee \boldsymbol{X}d)$

  $\mathcal{G} = \{ \boldsymbol{G}(a \vee \boldsymbol{F}b) \}$

  Reduces to checking $\boldsymbol{X}d$
- Reduces to checking the $\boldsymbol{G}$-free formula

  $\psi[\ \mathcal{G} \setminus \mathrm{tt}\ ,\ \overline{\mathcal{G}} \setminus \mathrm{ff}\ ]$
- Since the formula is $\boldsymbol{G}$-free, use the tracking automaton.

CHECKING 1.

- „$\mathcal{G}$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”

CHECKING 1.

- „$\mathcal{G}$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”

- Example: $\psi = \boldsymbol{F}c \wedge \boldsymbol{GF}(a \wedge (\boldsymbol{G}b \vee \boldsymbol{FG}c))$

$$\mathcal{G} = \{\, \boldsymbol{G}b \;,\; \boldsymbol{GF}\big(a \wedge (\boldsymbol{G}b \vee \boldsymbol{FG}c)\big) \,\}$$

reduces to checking $\boldsymbol{G}b \wedge \boldsymbol{GF}a \equiv \boldsymbol{G}(b \wedge \boldsymbol{F}a)$

CHECKING 1.

- „$\mathcal{G}$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”
- Example: $\psi = \boldsymbol{F}c \wedge \boldsymbol{GF}(a \wedge (\boldsymbol{G}b \vee \boldsymbol{FG}c))$

  $\mathcal{G} = \{ \boldsymbol{G}b \ , \ \boldsymbol{GF}(a \wedge (\boldsymbol{G}b \vee \boldsymbol{FG}c)) \}$

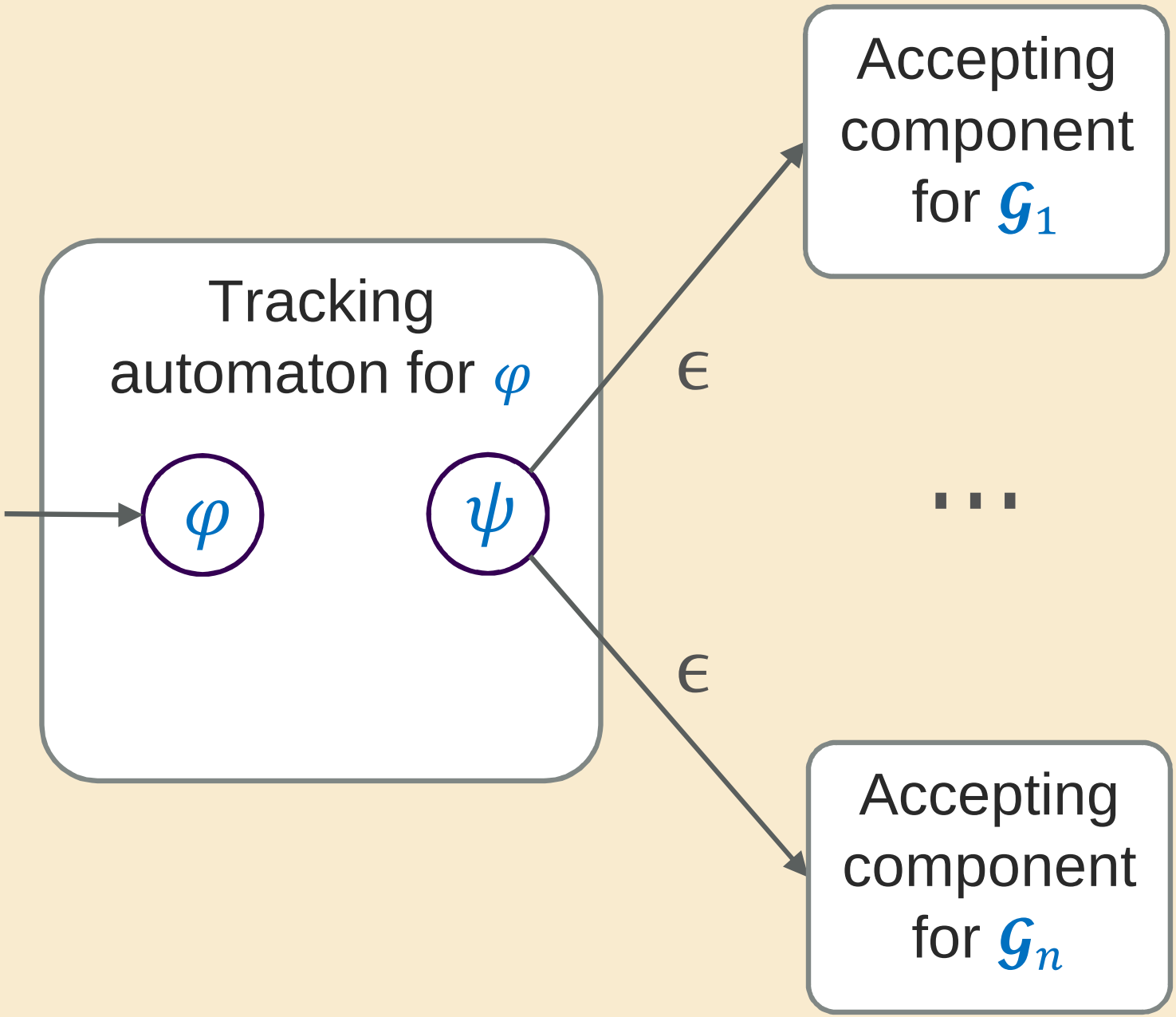
  reduces to checking $\boldsymbol{G}b \wedge \boldsymbol{GF}a \equiv \boldsymbol{G}(b \wedge \boldsymbol{F}a)$
- Reduces to checking a formula $\boldsymbol{G}\rho$ where ρ is $\boldsymbol{G}$-free.

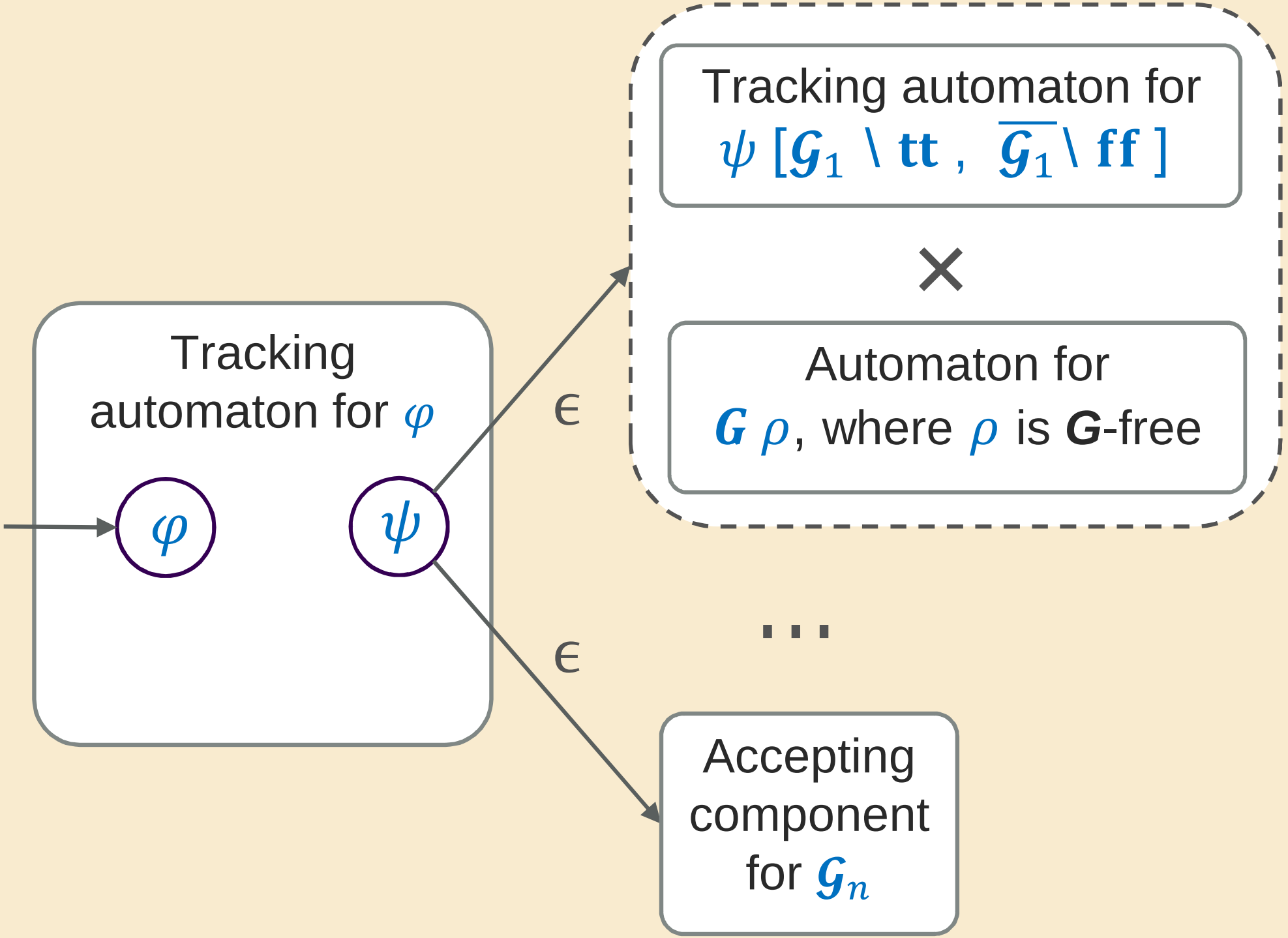
CHECKING 1.

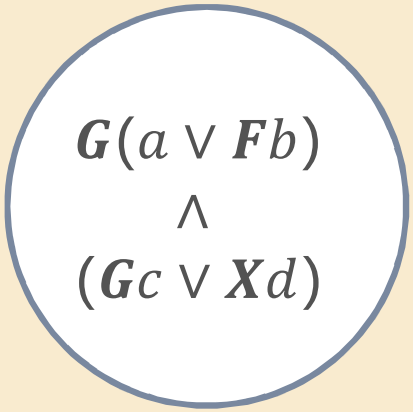
- „$\mathcal{G}$ holds even if no other $\boldsymbol{G}$-subformula of ψ ever becomes true”
- Example: $\psi = \boldsymbol{F}c \wedge \boldsymbol{GF}(a \wedge (\boldsymbol{G}b \vee \boldsymbol{FG}c))$

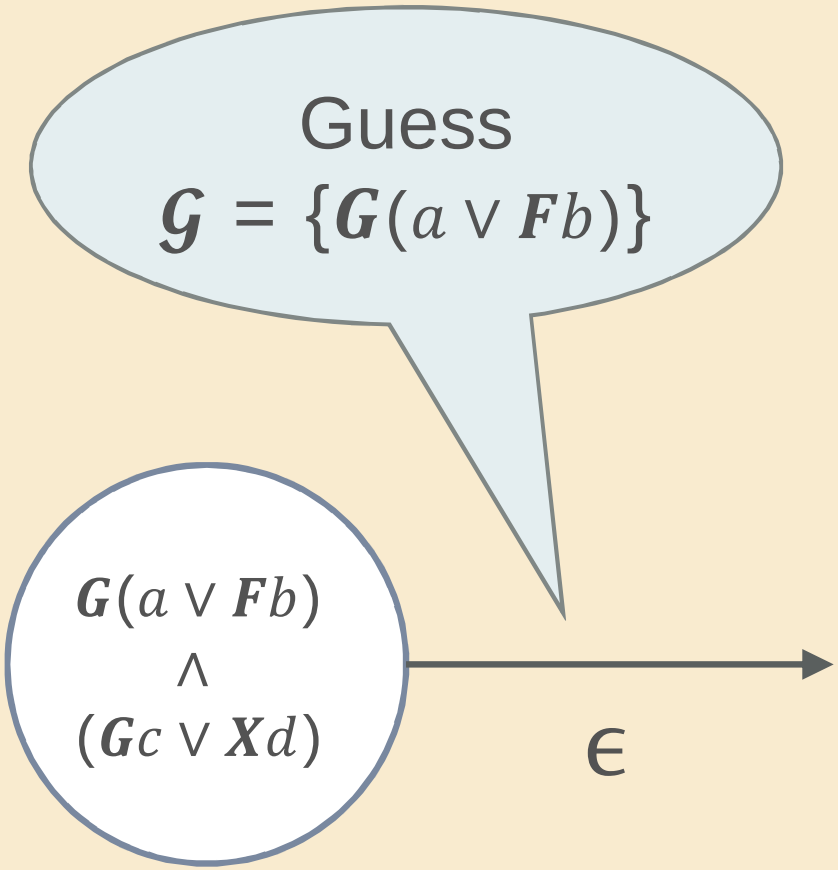
  $\mathcal{G} = \{\, \boldsymbol{G}b \;,\; \boldsymbol{GF}\big(a \wedge (\boldsymbol{G}b \vee \boldsymbol{FG}c)\big) \,\}$

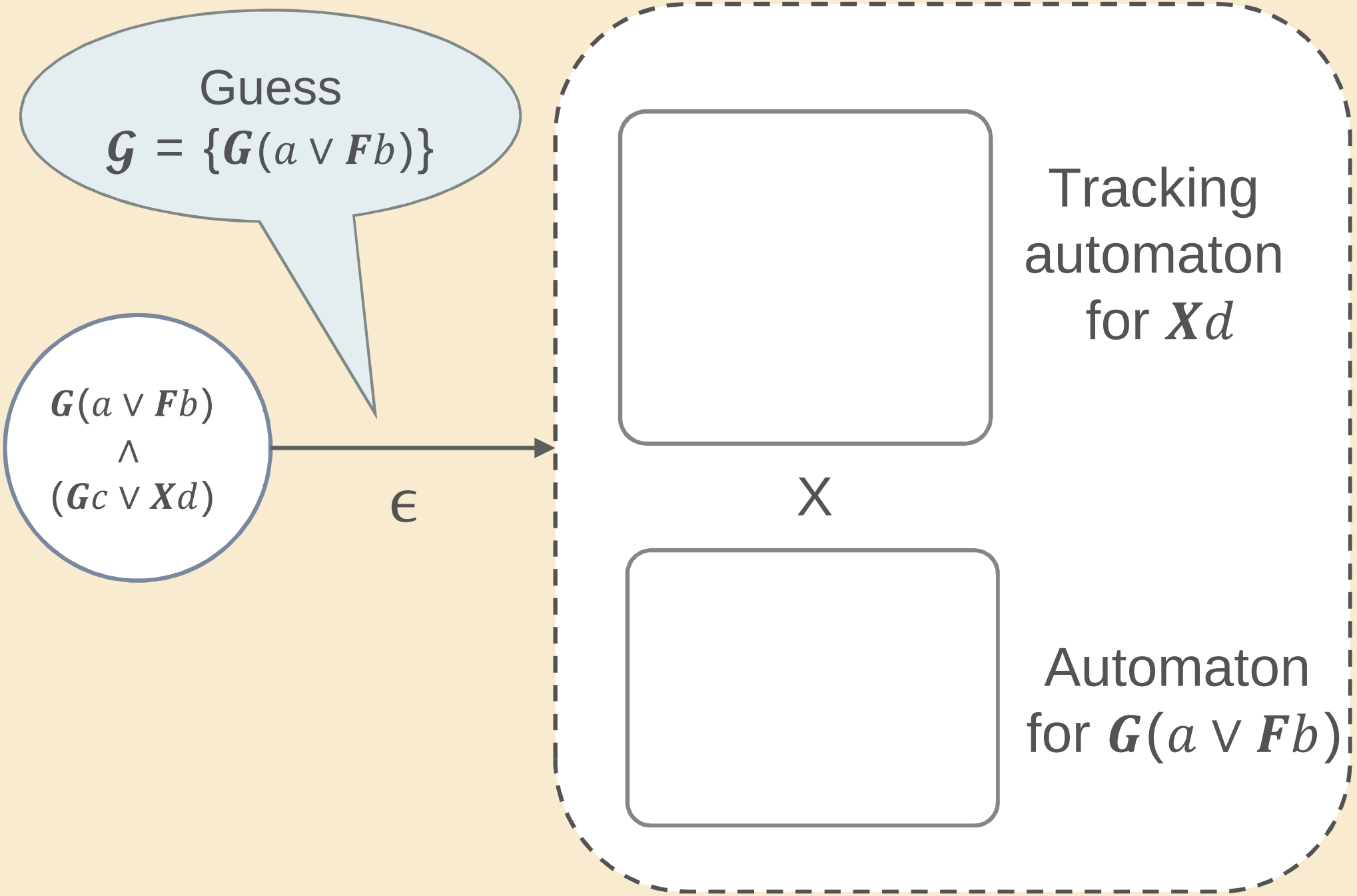
  reduces to checking $\boldsymbol{G}b \wedge \boldsymbol{GF}a \equiv \boldsymbol{G}(b \wedge \boldsymbol{F}a)$
- Reduces to checking a formula $\boldsymbol{G}\rho$ where ρ is $\boldsymbol{G}$-free.
- So we need DBAs for formulas $\boldsymbol{G}\rho$ where ρ is $\boldsymbol{G}$-free.

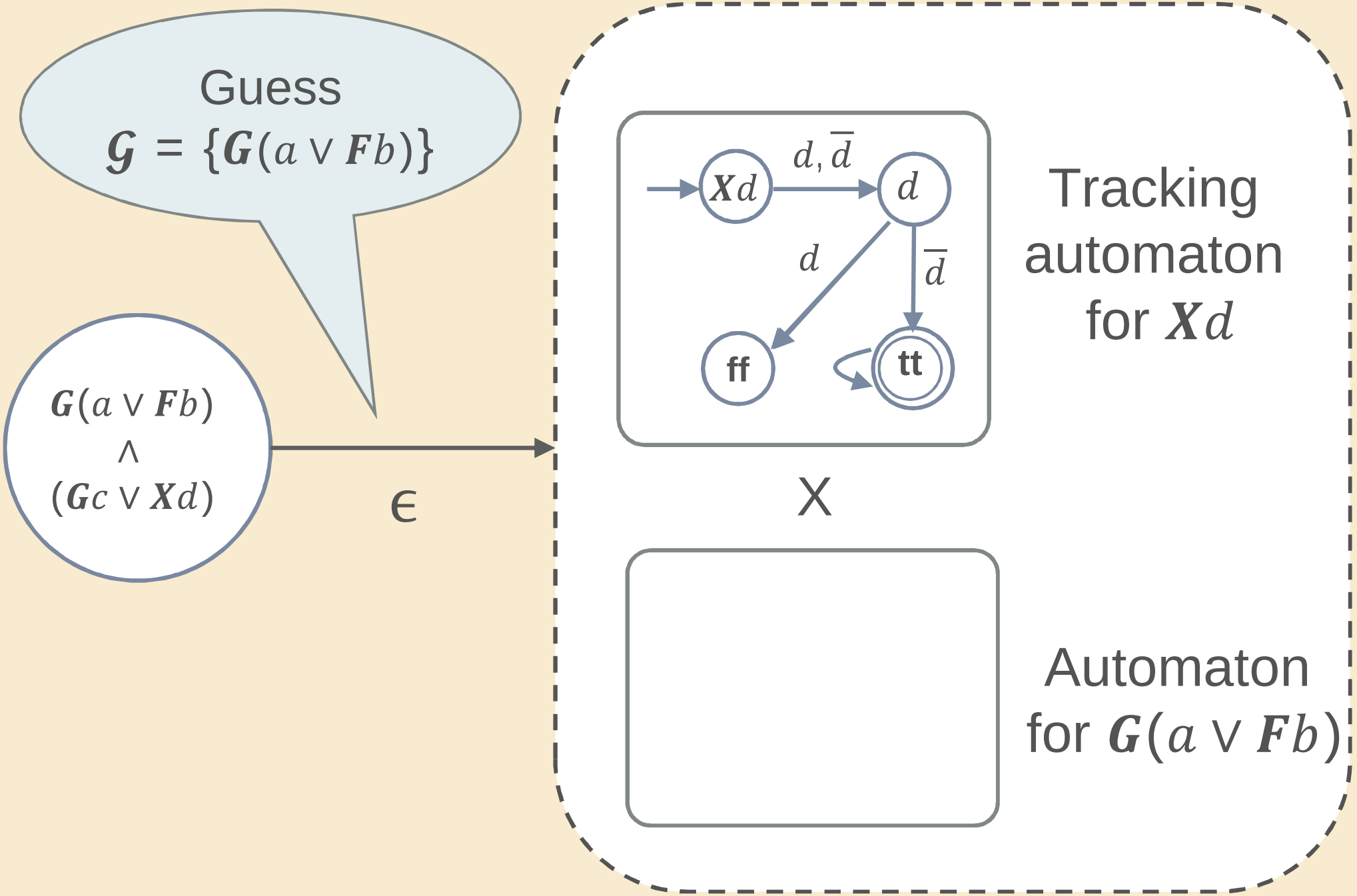

Accepting component for $\mathcal{G}_1$
Tracking automaton for φ
φ
ψ
ϵ
...
ϵ
Accepting component for $\mathcal{G}_n$

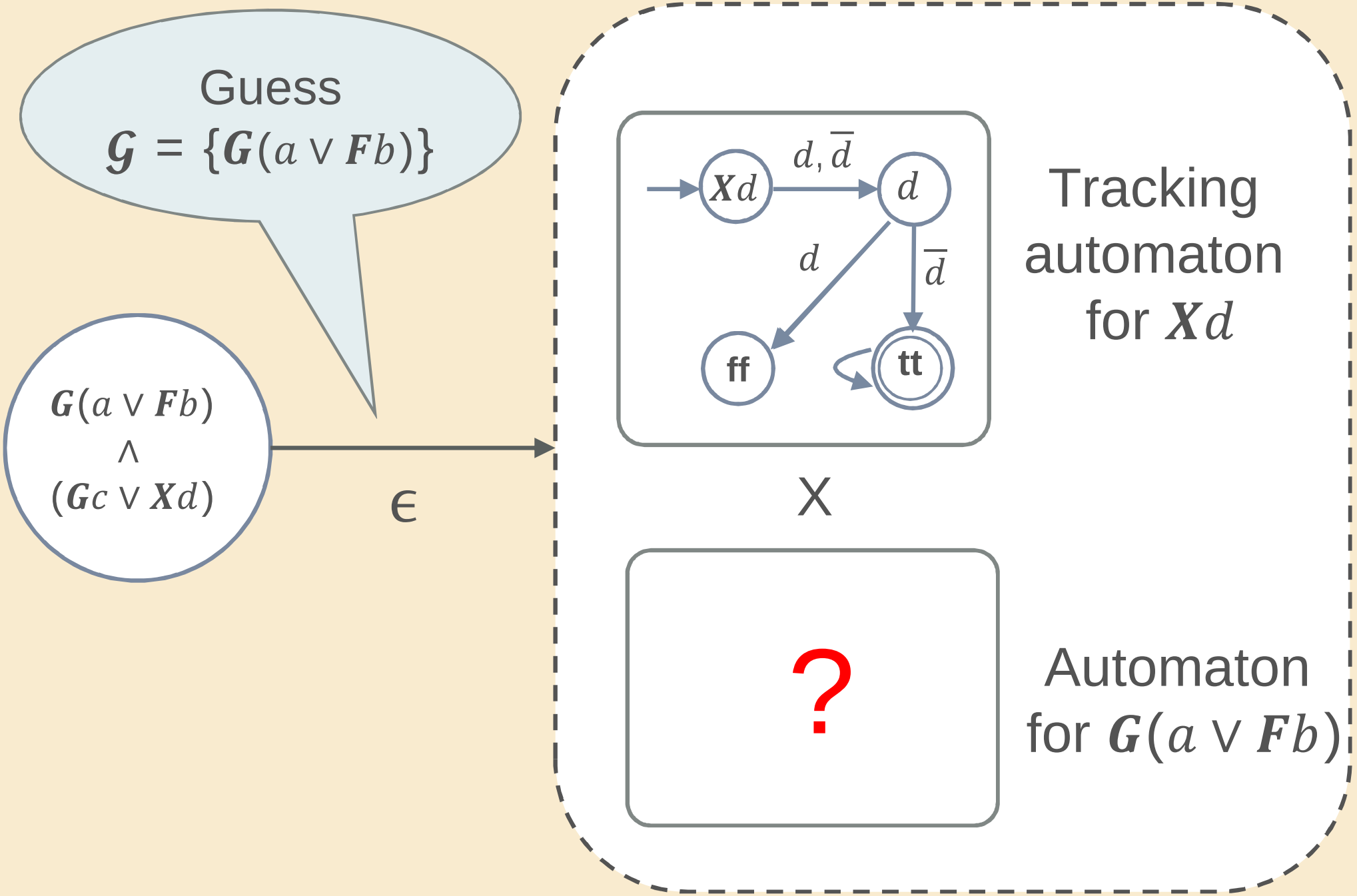

Tracking automaton for ψ [$\mathcal{G}_1$ \ tt , $\overline{\mathcal{G}_1}$ \ ff]
×
Automaton for $\boldsymbol{G}\ \rho$, where ρ is $\boldsymbol{G}$-free
Tracking automaton for φ
φ
ψ
ϵ
ϵ
...
Accepting component for $\mathcal{G}_n$


$\boldsymbol{G}(a \vee \boldsymbol{F}b)$
$\wedge$
$(\boldsymbol{G}c \vee \boldsymbol{X}d)$


Guess
$\mathcal{G} = \{\boldsymbol{G}(a \vee \boldsymbol{F}b)\}$
$\boldsymbol{G}(a \vee \boldsymbol{F}b)$
$\wedge$
$(\boldsymbol{G}c \vee \boldsymbol{X}d)$
$\in$


Guess
$\mathcal{G} = \{\boldsymbol{G}(a \vee \boldsymbol{F}b)\}$
$\boldsymbol{G}(a \vee \boldsymbol{F}b)$
$\wedge$
$(\boldsymbol{G}c \vee \boldsymbol{X}d)$
ϵ
Tracking automaton for $\boldsymbol{X}d$
X
Automaton for $\boldsymbol{G}(a \vee \boldsymbol{F}b)$

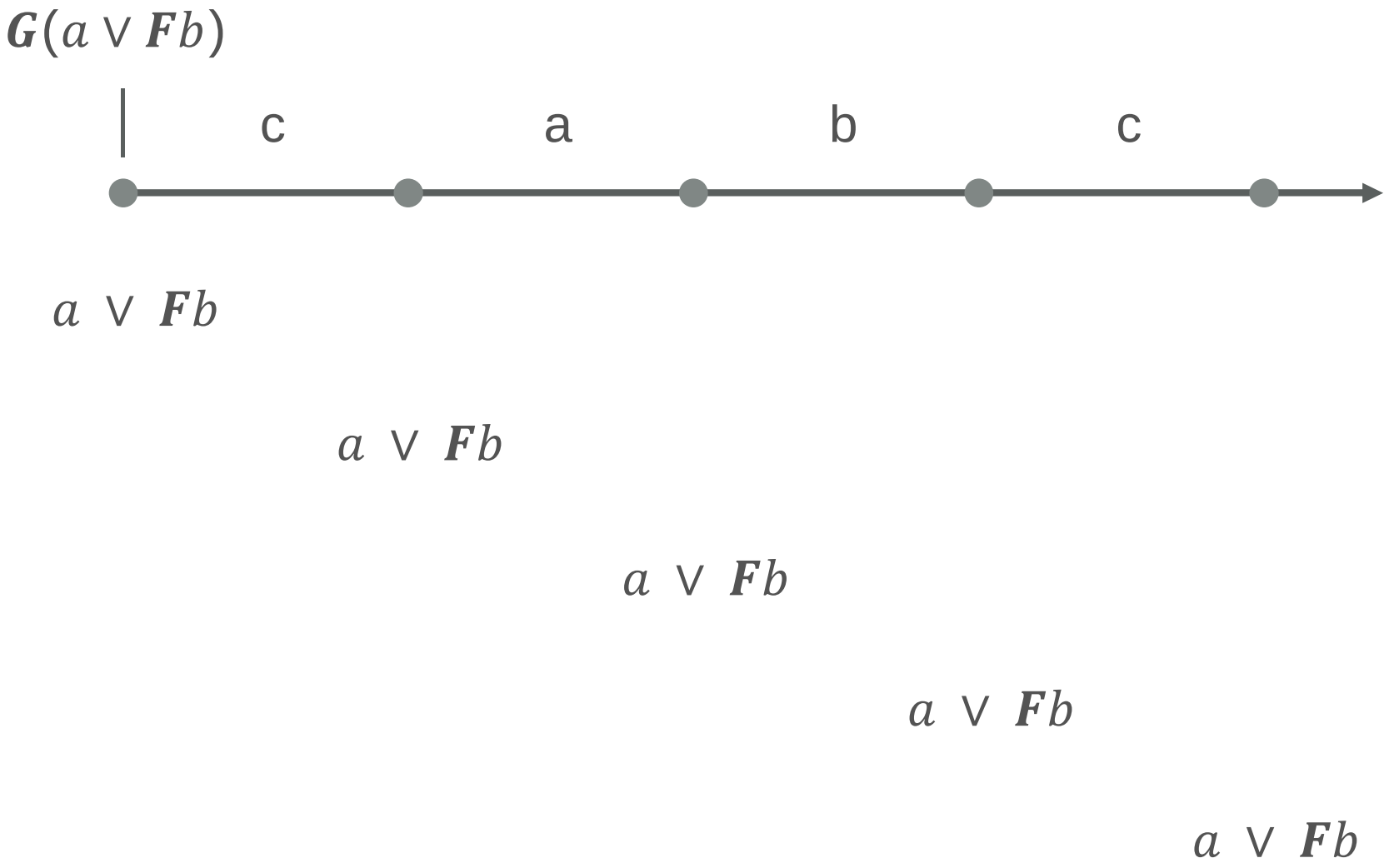

Guess
$\mathcal{G} = \{\boldsymbol{G}(a \vee \boldsymbol{F}b)\}$
$\boldsymbol{G}(a \vee \boldsymbol{F}b)$
$\wedge$
$(\boldsymbol{G}c \vee \boldsymbol{X}d)$
$\in$
$\boldsymbol{X}d$
$d, \overline{d}$
d
d
$\overline{d}$
ff
tt
Tracking automaton for $\boldsymbol{X}d$
X
Automaton for $\boldsymbol{G}(a \vee \boldsymbol{F}b)$


Guess
$\mathcal{G} = \{\boldsymbol{G}(a \vee \boldsymbol{F}b)\}$
$\boldsymbol{G}(a \vee \boldsymbol{F}b)$
$\wedge$
$(\boldsymbol{G}c \vee \boldsymbol{X}d)$
$\in$
$\boldsymbol{X}d$
$d, \overline{d}$
d
d
$\overline{d}$
ff
tt
Tracking automaton for $\boldsymbol{X}d$
X
?
Automaton for $\boldsymbol{G}(a \vee \boldsymbol{F}b)$

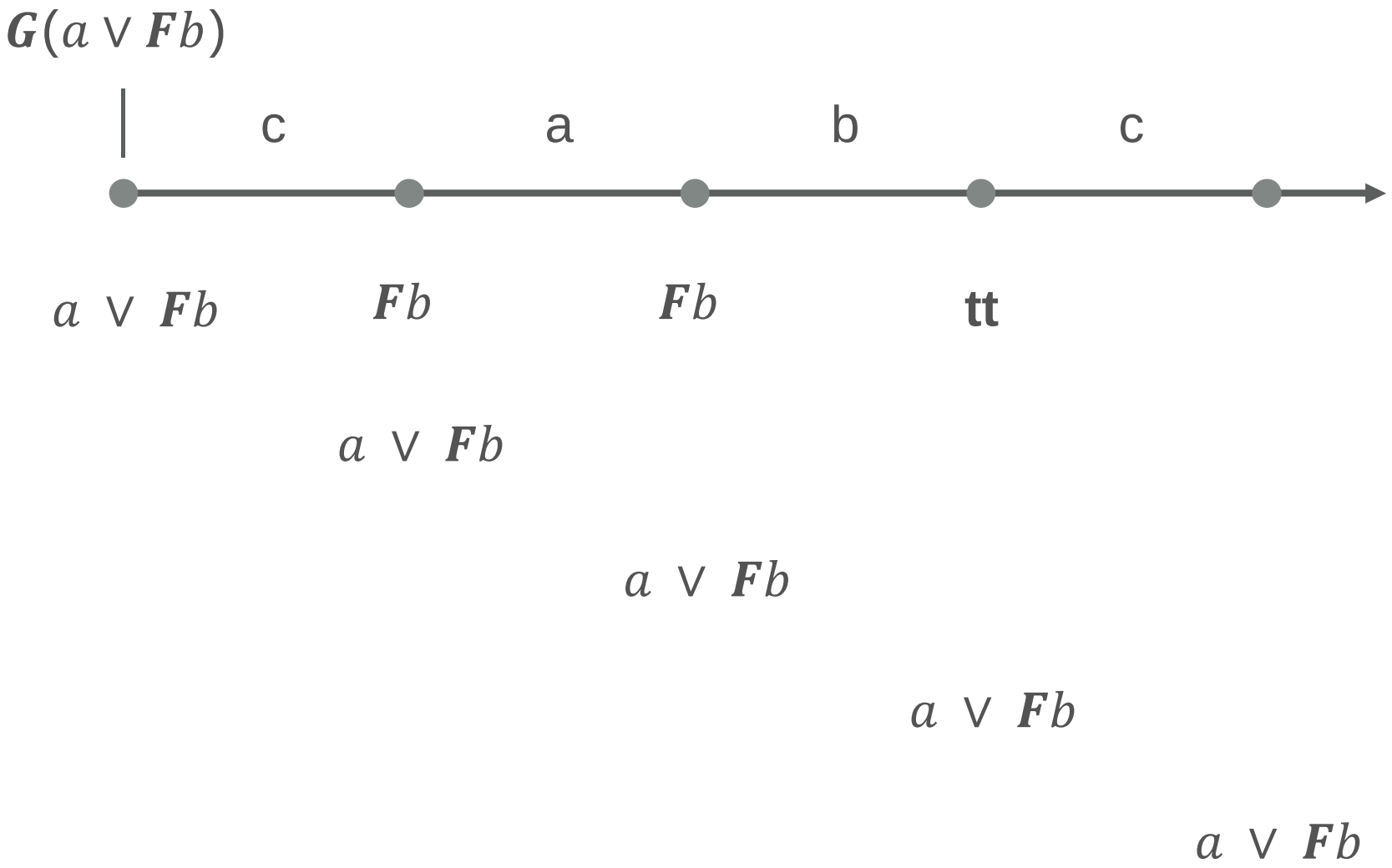
A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$

$\boldsymbol{G}(a \vee \boldsymbol{F}b)$

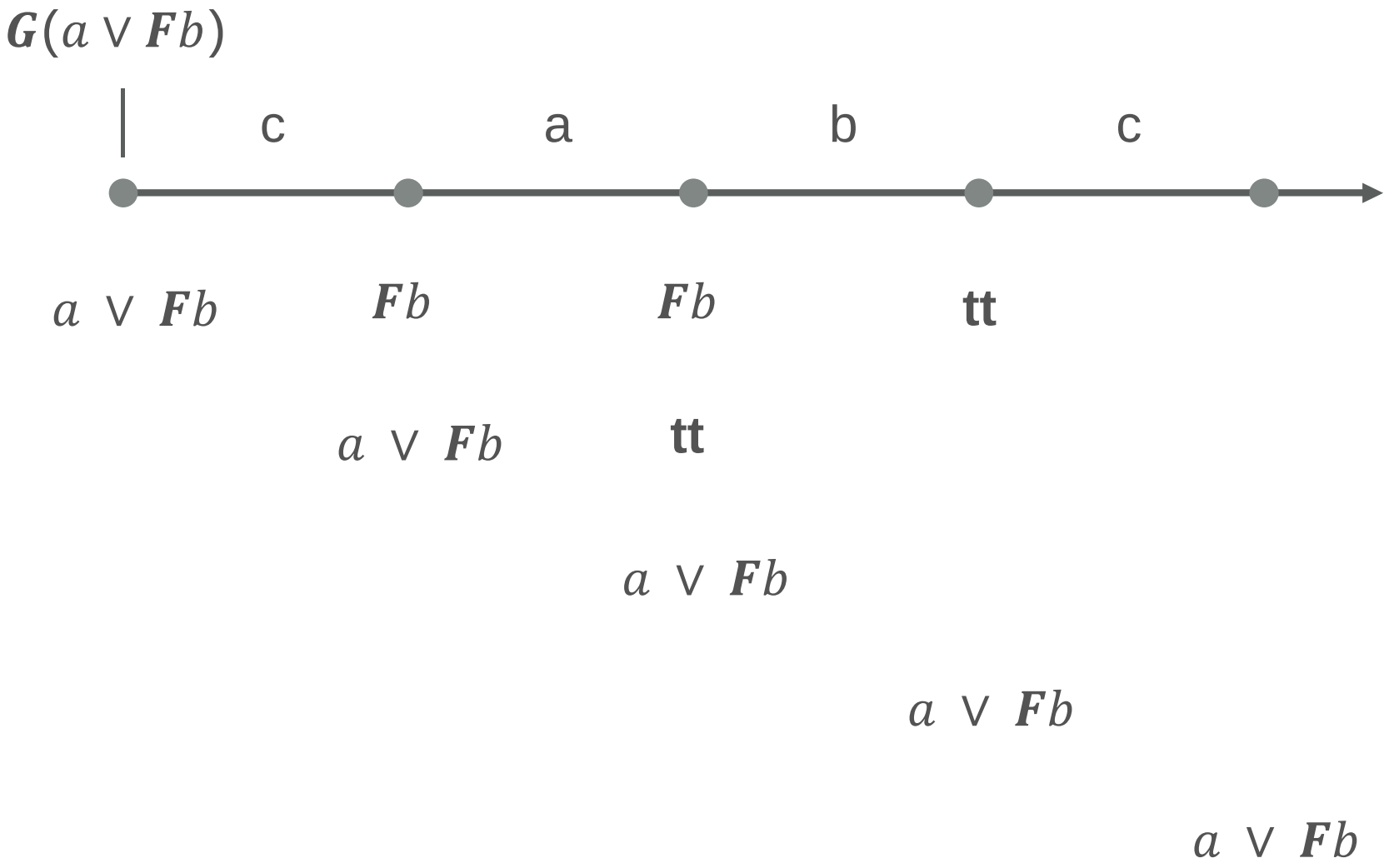
A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$



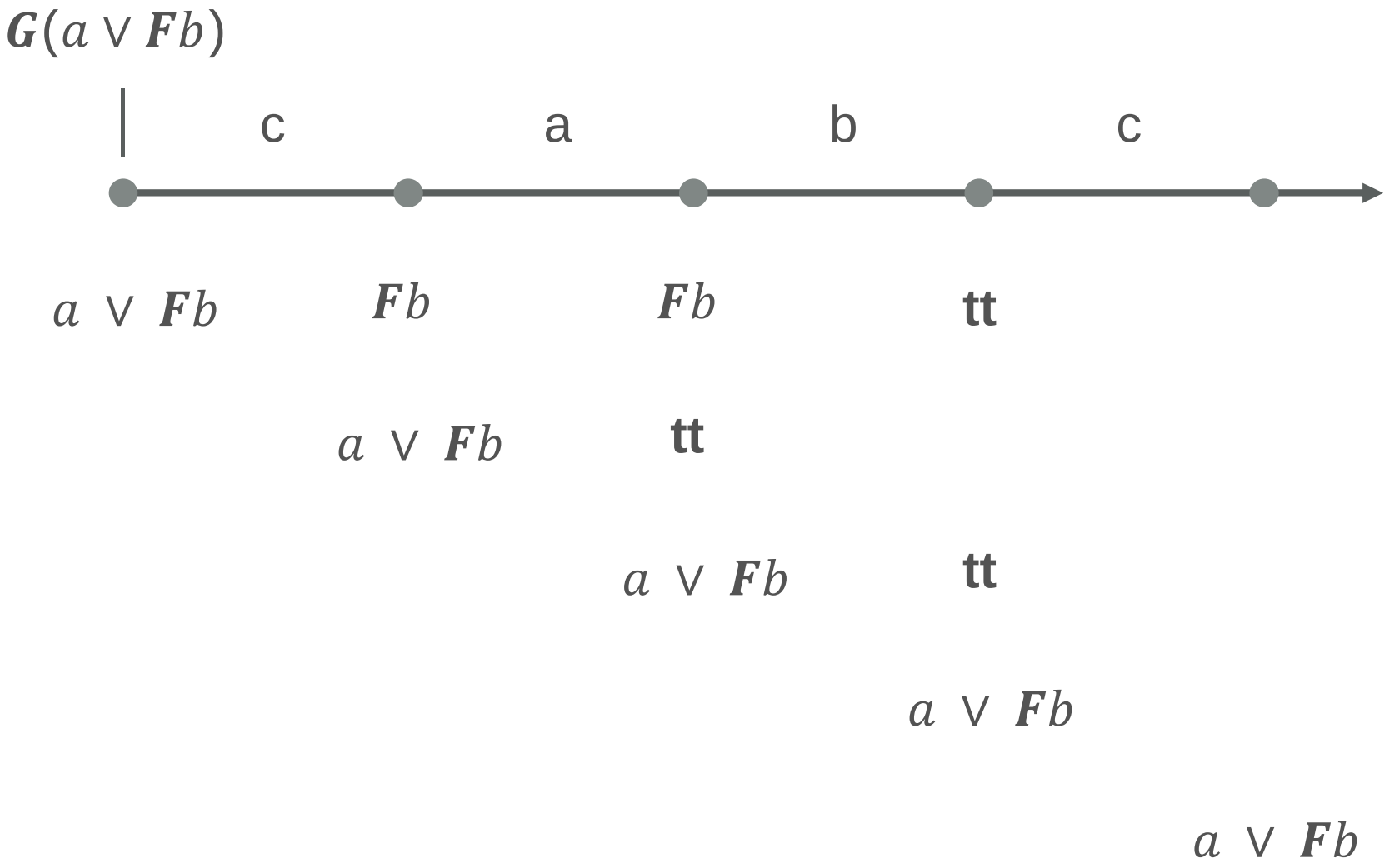
A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$



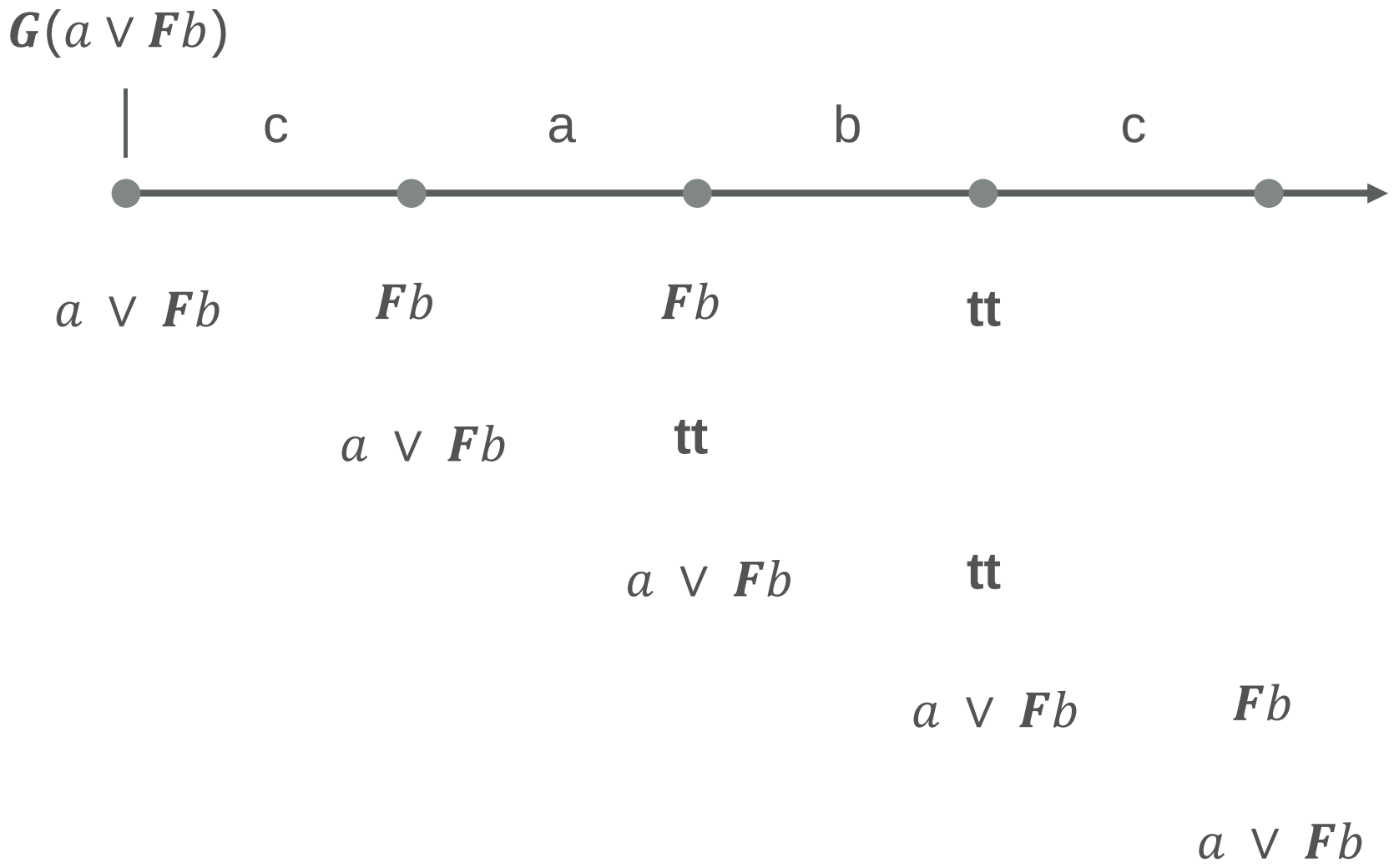
A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$



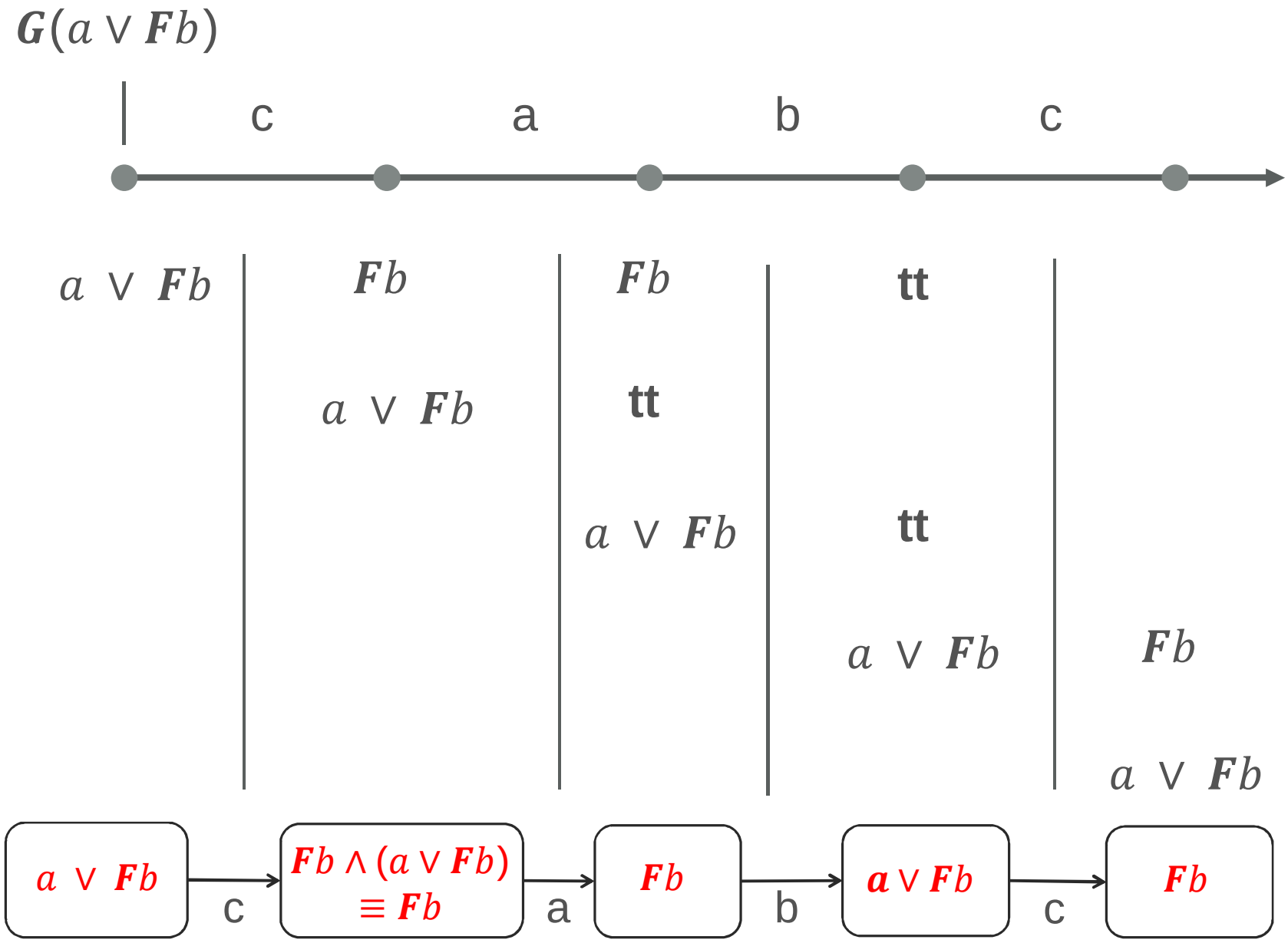
A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$



A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$



A DBA FOR $\boldsymbol{G}(a \vee \boldsymbol{F}b)$

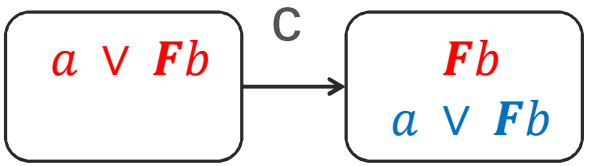


$a \vee \boldsymbol{F}b$

$a \vee \boldsymbol{F}b$

$a \vee \boldsymbol{F}b$ $\boldsymbol{F}b$

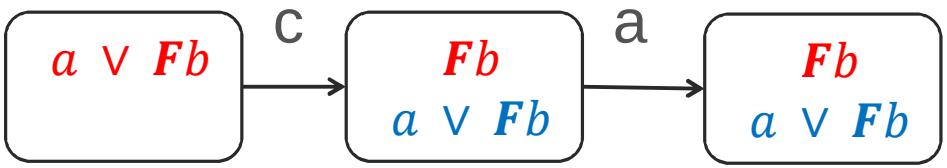
$a \vee \boldsymbol{F}b$

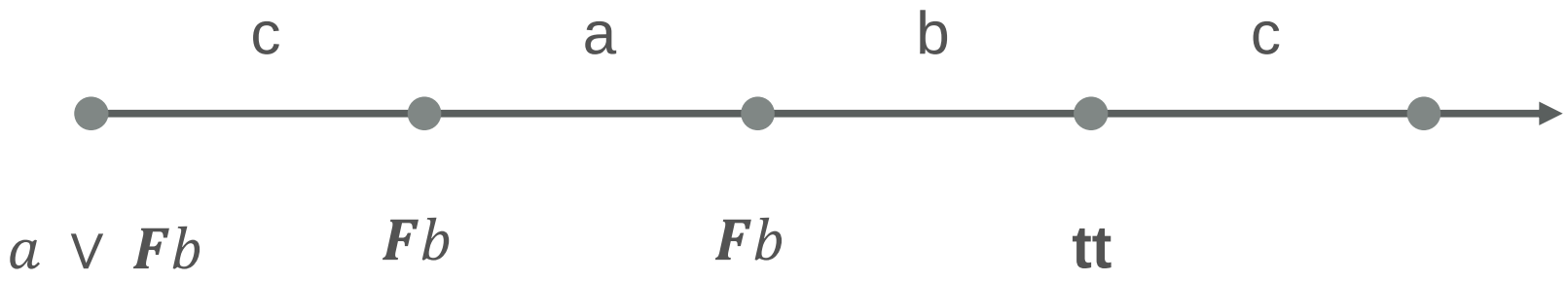


$a \vee \boldsymbol{F}b$ $\quad$ $\boldsymbol{F}b$ $\quad$ $\boldsymbol{F}b$

$a \vee \boldsymbol{F}b$ $\quad$ **tt**

$a \vee \boldsymbol{F}b$

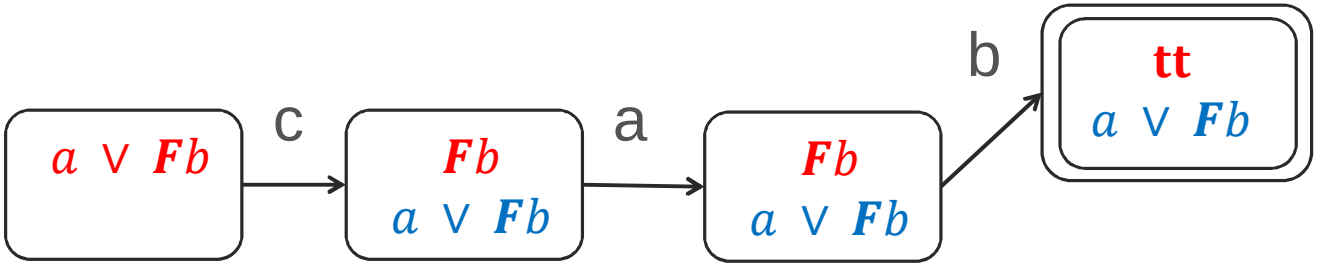




$a \vee Fb$ tt

$a \vee Fb$ tt

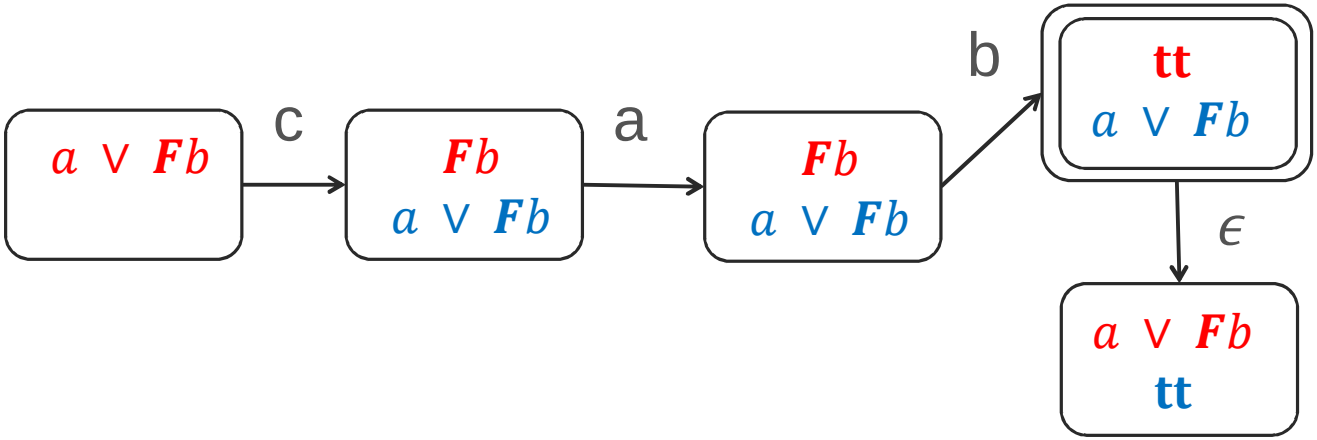
$a \vee Fb$



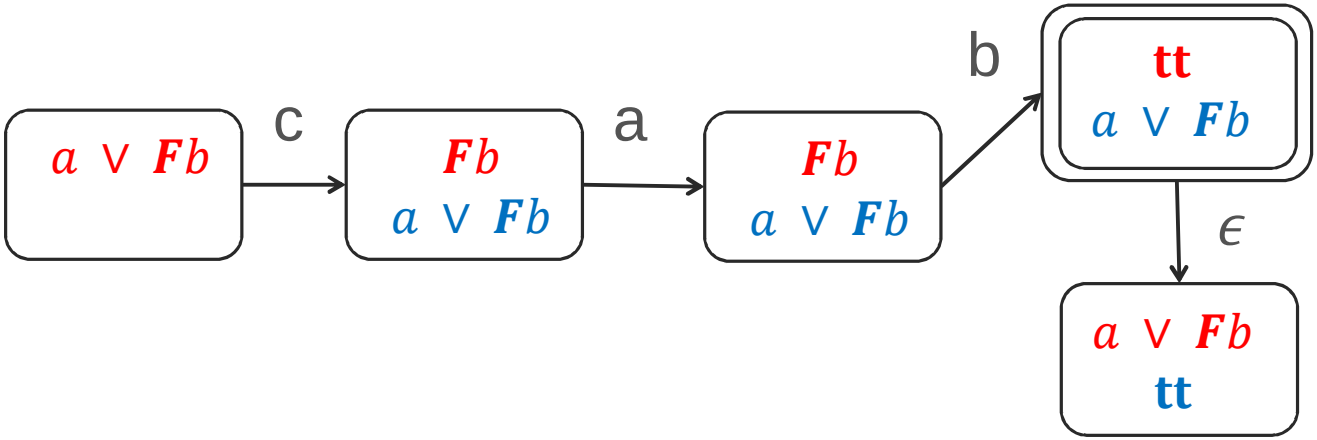
$a \vee \boldsymbol{F}b$ **tt**

$a \vee \boldsymbol{F}b$ **tt**

$a \vee \boldsymbol{F}b$

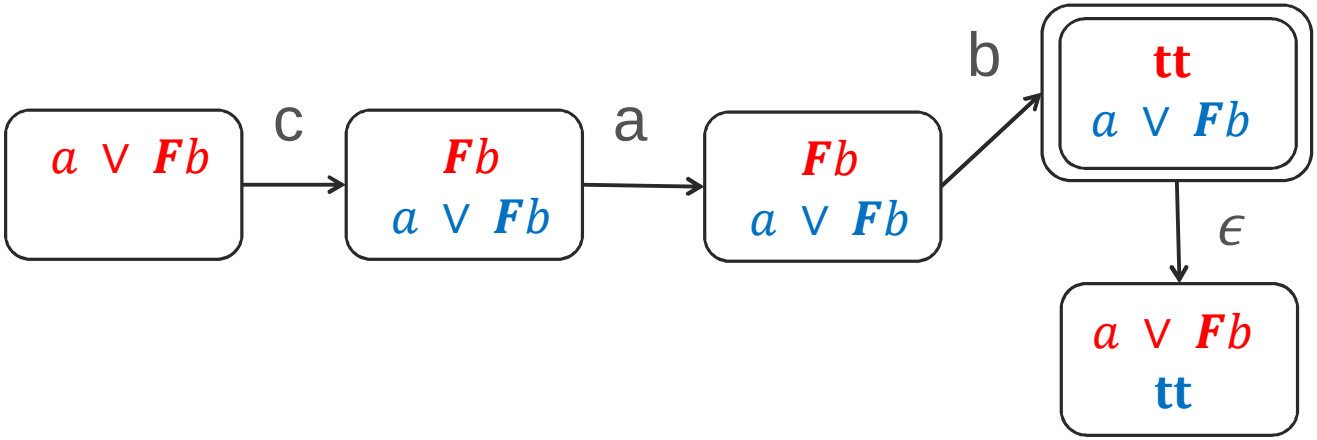


$a \vee \boldsymbol{F}b$

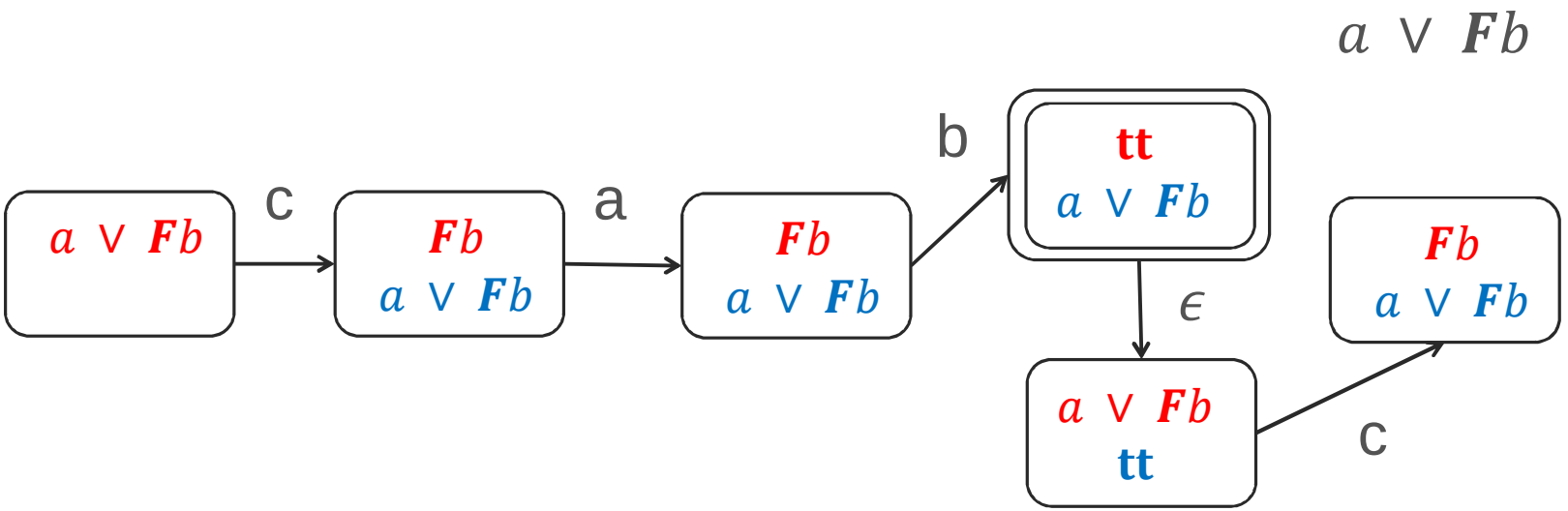


$a \vee \boldsymbol{F}b$ $\boldsymbol{F}b$

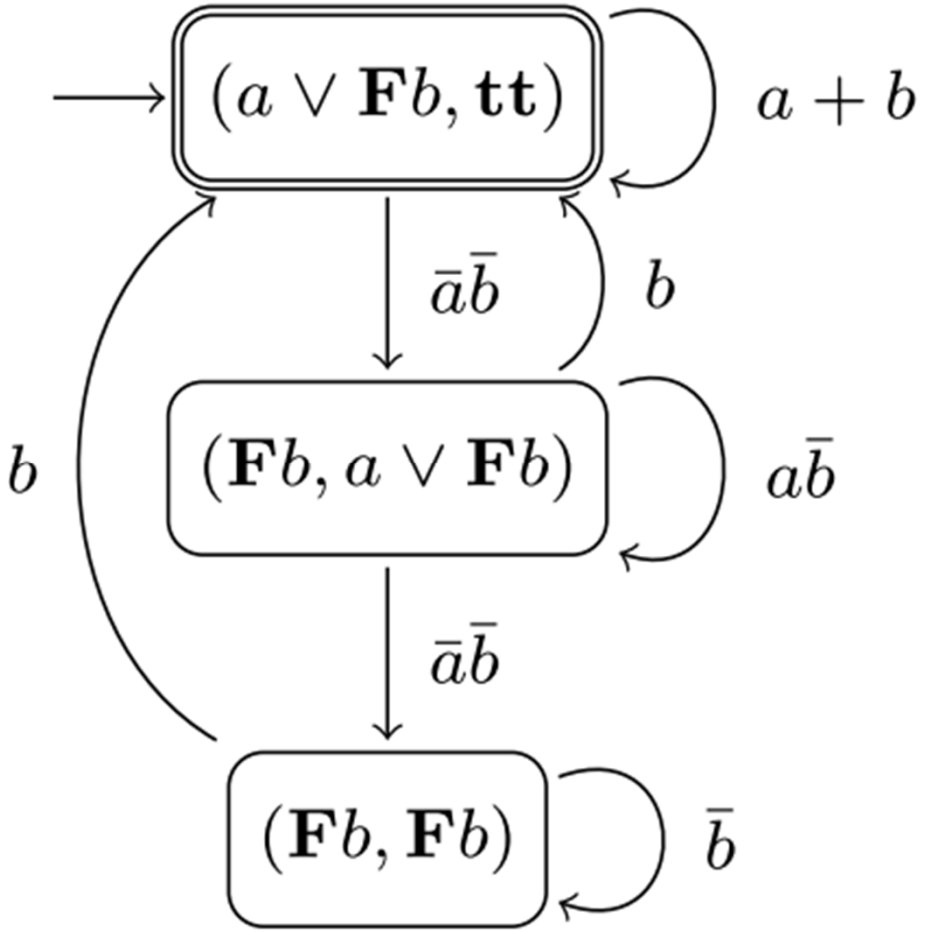
$a \vee \boldsymbol{F}b$



$a \vee \boldsymbol{F}b$ $\boldsymbol{F}b$

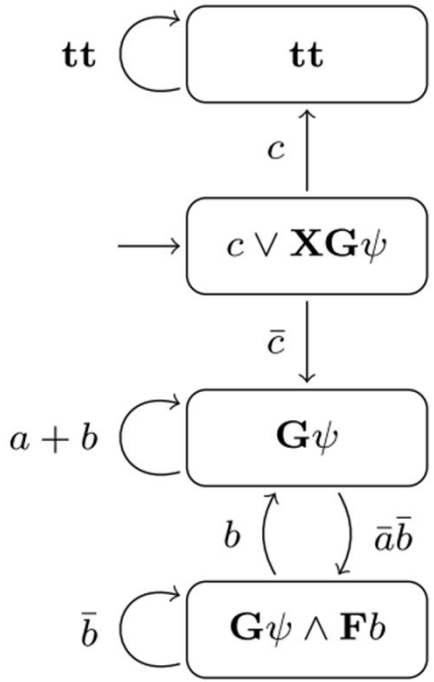


DBA FOR $G(a \vee Fb)$



COMPLETE LDBA FOR $\varphi = c \vee XG(a \vee Fb)$

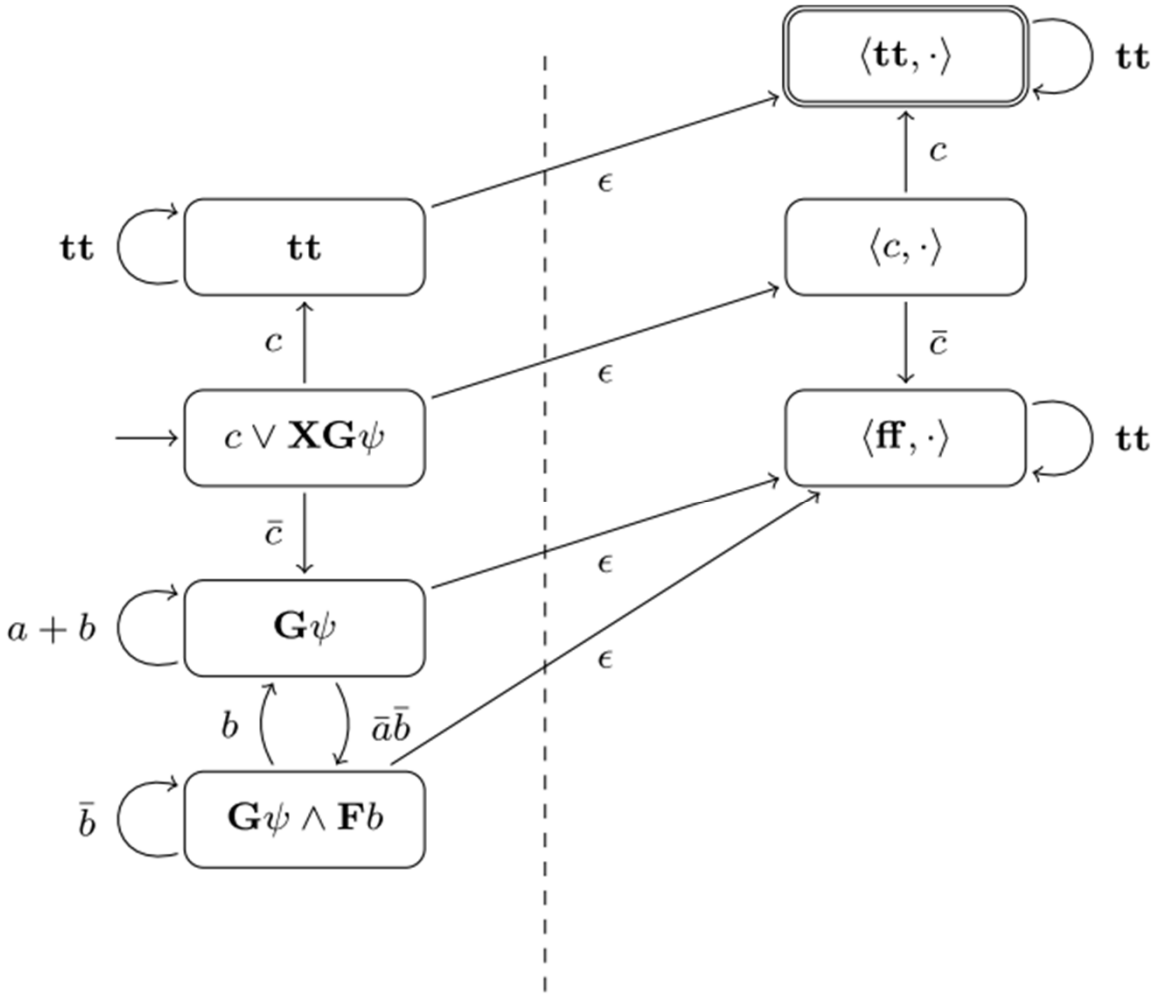
1. Tracking DBA for φ (abbr. $\psi \coloneqq a \vee Fb$)



COMPLETE LDBA FOR $\varphi = c \vee XG(a \vee Fb)$

1. Tracking DBA for φ (abbr. $\psi \coloneqq a \vee Fb$)

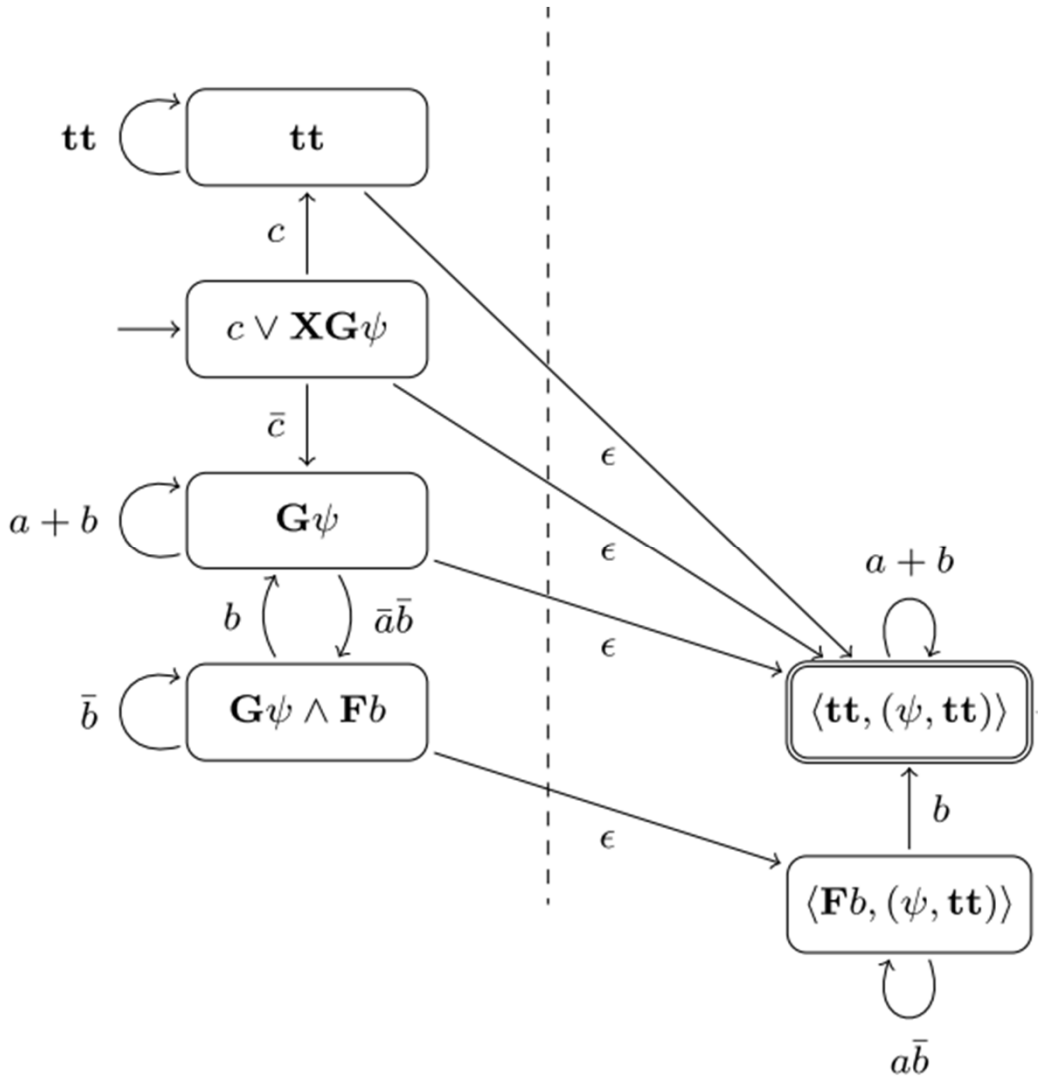
2. For every set $\mathcal{G}$ add a $\mathcal{G}$-jump to the product of the automata checking $\mathcal{G}$ and the $\mathcal{G}$ –remainder



COMPLETE LDBA FOR $\varphi = c \vee XG(a \vee Fb)$

1. Tracking DBA for φ (abbr. $\psi \coloneqq a \vee Fb$)

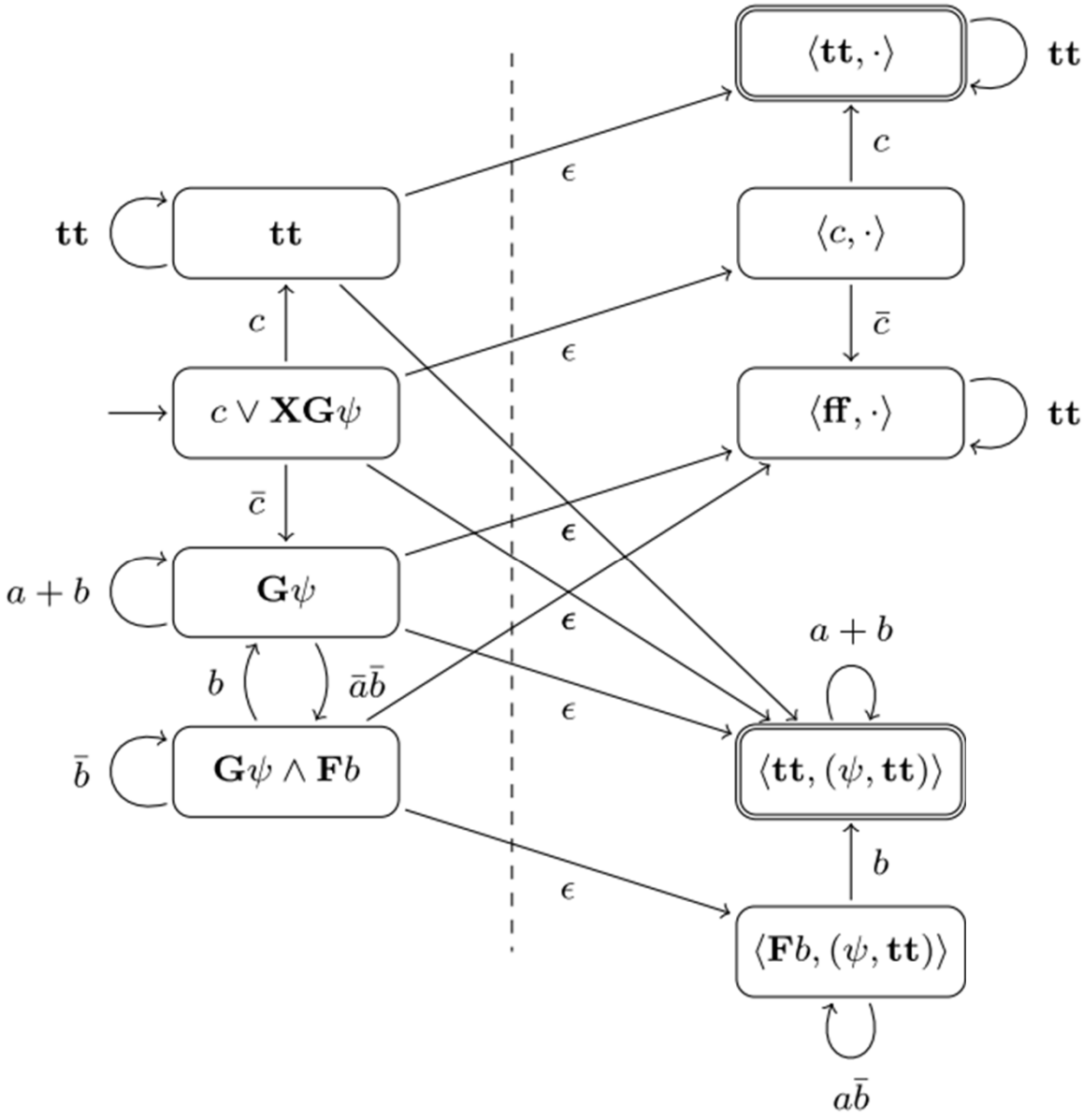
2. For every set $\mathcal{G}$ add a $\mathcal{G}$-jump to the product of the automata checking $\mathcal{G}$ and the $\mathcal{G}$ –remainder



COMPLETE LDBA FOR $\varphi = c \vee XG(a \vee Fb)$

1. Tracking DBA for φ (abbr. $\psi \coloneqq a \vee Fb$)

2. For every set $\mathcal{G}$ add a $\mathcal{G}$-jump to the product of the automata checking $\mathcal{G}$ and the $\mathcal{G}$ –remainder



UPPER BOUND ON LDBA SIZE

- Theorem: Every formula obtained by „tracking φ“ is a positive boolean combination of subformulas of φ.

$$c \vee \boldsymbol{XG}(a \vee \boldsymbol{F}b) \overset{b}{\rightarrow} \boldsymbol{G}(a \vee \boldsymbol{F}b) \overset{c}{\rightarrow} \boldsymbol{G}(a \vee \boldsymbol{F}b) \wedge \boldsymbol{F}b$$

UPPER BOUND ON LDBA SIZE

- Theorem: Every formula obtained by „tracking φ“ is a positive boolean combination of subformulas of φ.

$$c \vee \boldsymbol{XG}(a \vee \boldsymbol{F}b) \overset{b}{\rightarrow} \boldsymbol{G}(a \vee \boldsymbol{F}b) \overset{c}{\rightarrow} \boldsymbol{G}(a \vee \boldsymbol{F}b) \wedge \boldsymbol{F}b$$

- Corollary: for a formula of length n there are at most 2^{2^n} „tracking formulas” up to equivalence, even if we leave temporal operators uninterpreted.

$$\boldsymbol{F}a \wedge (\boldsymbol{F}a \vee \boldsymbol{G}b) =_P \boldsymbol{F}a \quad \boldsymbol{F}a \vee \boldsymbol{G}a \neq_P \boldsymbol{F}a$$

UPPER BOUND ON LDBA SIZE

- Theorem: Every formula obtained by „tracking φ“ is a positive boolean combination of subformulas of φ.

$$c \vee \boldsymbol{XG}(a \vee \boldsymbol{F}b) \overset{b}{\rightarrow} \boldsymbol{G}(a \vee \boldsymbol{F}b) \overset{c}{\rightarrow} \boldsymbol{G}(a \vee \boldsymbol{F}b) \wedge \boldsymbol{F}b$$

- Corollary: for a formula of length n there are at most 2^{2^n} „tracking formulas” up to equivalence, even if we leave temporal operators uninterpreted.

$$\boldsymbol{F}a \wedge (\boldsymbol{F}a \vee \boldsymbol{G}b) =_P \boldsymbol{F}a \quad \boldsymbol{F}a \vee \boldsymbol{G}a \neq_P \boldsymbol{F}a$$

- This allows us to derive an upper bound on the size of the LDBA

UPPER BOUND ON LDBA SIZE

Part	Size
Initial Component	2^{2^n}
G-Monitor	$2^{2^{n+1}}$
Accepting Component	$2^{2^{O(n)}}$
Total	$2^{2^{O(n)}}$

LDBA SIZE IN PRACTICE

$$\bigwedge_{i=1}^{j}(\mathbf{GF}a_i) \implies \bigwedge_{i=1}^{j}(\mathbf{GF}b_i)$$

$$k\colon \bigwedge_{i=1}^{k}(\mathbf{GF}a_i \vee \mathbf{FG}b_i)$$

$$f(0,j) = (\mathbf{GF}a_0)\mathbf{U}(\mathbf{X}^j b)$$

$$f(i+1,j) = (\mathbf{GF}a_{i+1})\mathbf{U}(\mathbf{G}f(i,j))$$

	LDBA	Safra (spot+ltl2dstar)	Rabinizer
$j=1$	3	5	2
$j=2$	4	17	3
$j=3$	5	49	4
$j=4$	6	129	5
$k=2$	5	4385	13
$k=3$	9	*	198
$f(0,0)$	5	5	5
$f(0,2)$	10	10	7
$f(0,4)$	16	12	9
$f(1,0)$	6	196	17
$f(1,2)$	28	109839	33
$f(1,4)$	58	*	70
$f(2,0)$	10	99793	41
$f(2,2)$	46	*	94
$f(2,4)$	92	*	139

LDBA SIZE IN PRACTICE

$$\bigwedge_{i=1}^{j}(\mathbf{GF}a_i) \implies \bigwedge_{i=1}^{j}(\mathbf{GF}b_i)$$

$$k\colon \bigwedge_{i=1}^{k}(\mathbf{GF}a_i \vee \mathbf{FG}b_i)$$

$$f(0,j) = (\mathbf{GF}a_0)\mathbf{U}(\mathbf{X}^j b)$$

$$f(i+1,j) = (\mathbf{GF}a_{i+1})\mathbf{U}(\mathbf{G}f(i,j))$$

	LDBA	Safra (spot+ltl2dstar)	Rabinizer
$j=1$	3	5	2
$j=2$	4	17	3
$j=3$	5	49	4
$j=4$	6	129	5
$k=2$	5	4385	13
$k=3$	9	*	198
$f(0,0)$	5	5	5
$f(0,2)$	10	10	7
$f(0,4)$	16	12	9
$f(1,0)$	6	196	17
$f(1,2)$	28	109839	33
$f(1,4)$	58	*	70
$f(2,0)$	10	99793	41
$f(2,2)$	46	*	94
$f(2,4)$	92	*	139

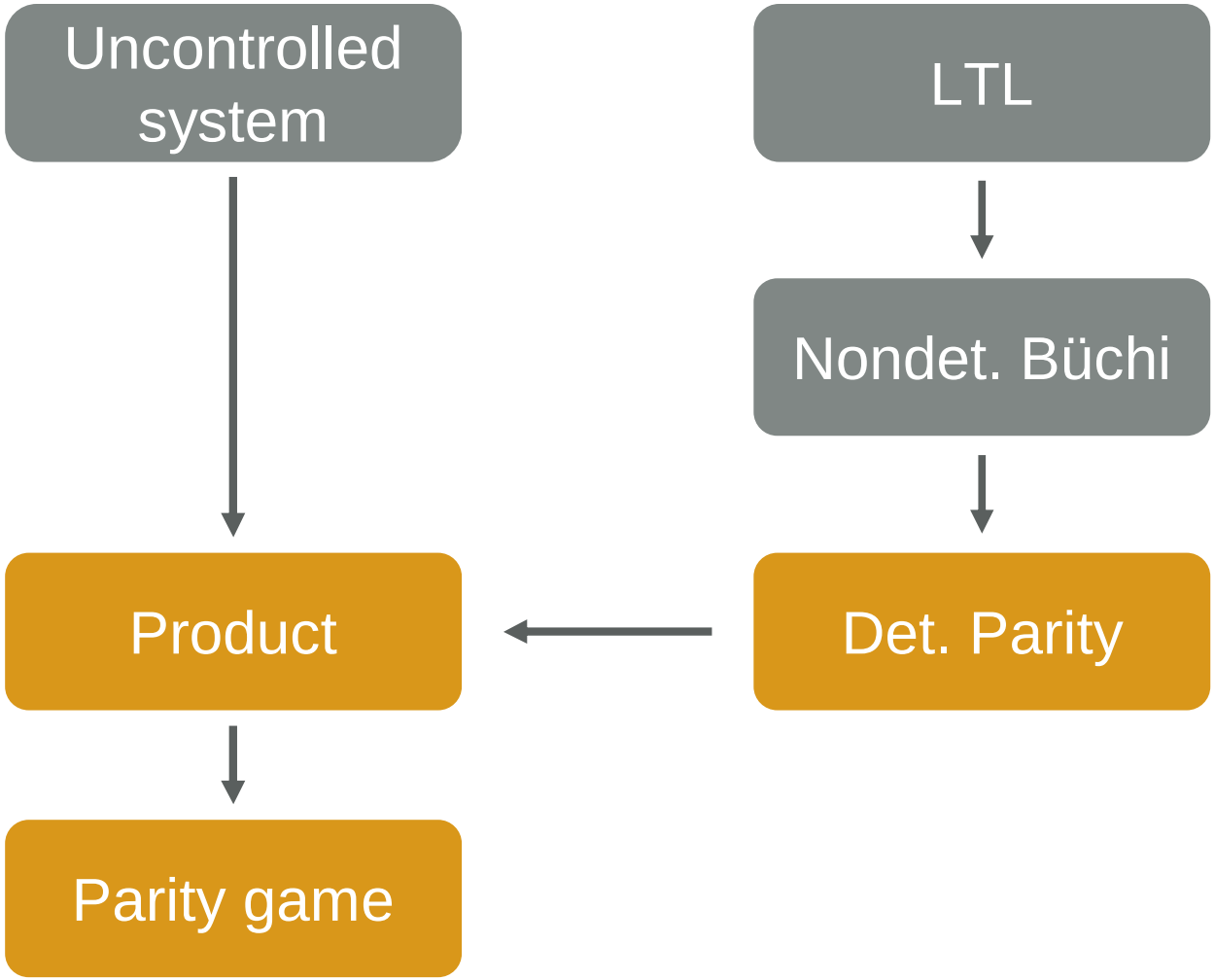
MODEL CHECKING RUNTIME
PNUELI-ZUCK MUTEX PROTOCOL

		#Clients	Our Imp. explicit	PRISM symb	PRISM+ Rabinizer symbolic	IscasMC explicit
(6)	$\mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}0 \vee \mathbf{FG}p2{\neq}0) \\ \wedge(\mathbf{GF}p2{=}0 \vee \mathbf{FG}p3{\neq}0) \\ \wedge(\mathbf{GF}p3{=}0 \vee \mathbf{FG}p1{\neq}0)\end{smallmatrix}]$	4	$< \mathbf{1}$	78	9	3
		5	**10**	1293	137	29
(7)	$\mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}0 \vee \mathbf{FG}p1{\neq}0) \\ \wedge(\mathbf{GF}p2{=}0 \vee \mathbf{FG}p2{\neq}0) \\ \wedge(\mathbf{GF}p3{=}0 \vee \mathbf{FG}p3{\neq}0)\end{smallmatrix}]$	4	$< \mathbf{1}$	$< \mathbf{1}$	61	2
		5	1	$< \mathbf{1}$	1077	27
(8)	$\mathbb{P}_{min=?}[\begin{smallmatrix}(\mathbf{GF}p1{\neq}10 \vee \mathbf{GF}p1{=}0 \vee \mathbf{FG}p1{=}1) \\ \wedge \mathbf{GF}p1{\neq}0 \wedge \mathbf{GF}p1{=}1\end{smallmatrix}]$	4	$< \mathbf{1}$	8	8	1
		5	**1**	145	190	16
(9)	$\mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{G}p1{\neq}10 \vee \mathbf{G}p2{\neq}10 \vee \mathbf{G}p3{\neq}10) \\ \wedge(\mathbf{FG}p1{\neq}1 \vee \mathbf{GF}p2{=}1 \vee \mathbf{GF}p3{=}1) \\ \wedge(\mathbf{FG}p2{\neq}1) \vee \mathbf{GF}p1{=}1 \vee \mathbf{GF}p3{=}1)\end{smallmatrix}]$	4	**5**	-	1195	8
		5	**99**	-	-	125
(10)	$\mathbb{P}_{min=?}[\begin{smallmatrix}\mathbf{FG}p1{\neq}0 \vee \mathbf{FG}p2{\neq}0 \\ \vee \mathbf{GF}p3{=}0 \vee (\mathbf{FG}p1{\neq}10 \\ \wedge \mathbf{GF}p2{=}10 \wedge \mathbf{GF}p3{=}10)\end{smallmatrix}]$	4	**1**	728	33	79
		5	**24**	-	486	-
(11)	$\mathbb{P}_{min=?}[f_{0,0}] = \mathbb{P}_{min=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}10)\mathbf{U} \\ (p2{=}10)\end{smallmatrix}]$	4	$< \mathbf{1}$	17	40	2
		5	**11**	257	715	23
(12)	$\mathbb{P}_{max=?}[f_{0,4}] = \mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}10)\mathbf{U} \\ (\mathbf{XXXX}p2{=}10)\end{smallmatrix}]$	4	$< \mathbf{1}$	3	$< \mathbf{1}$	1
		5	5	20	**2**	20

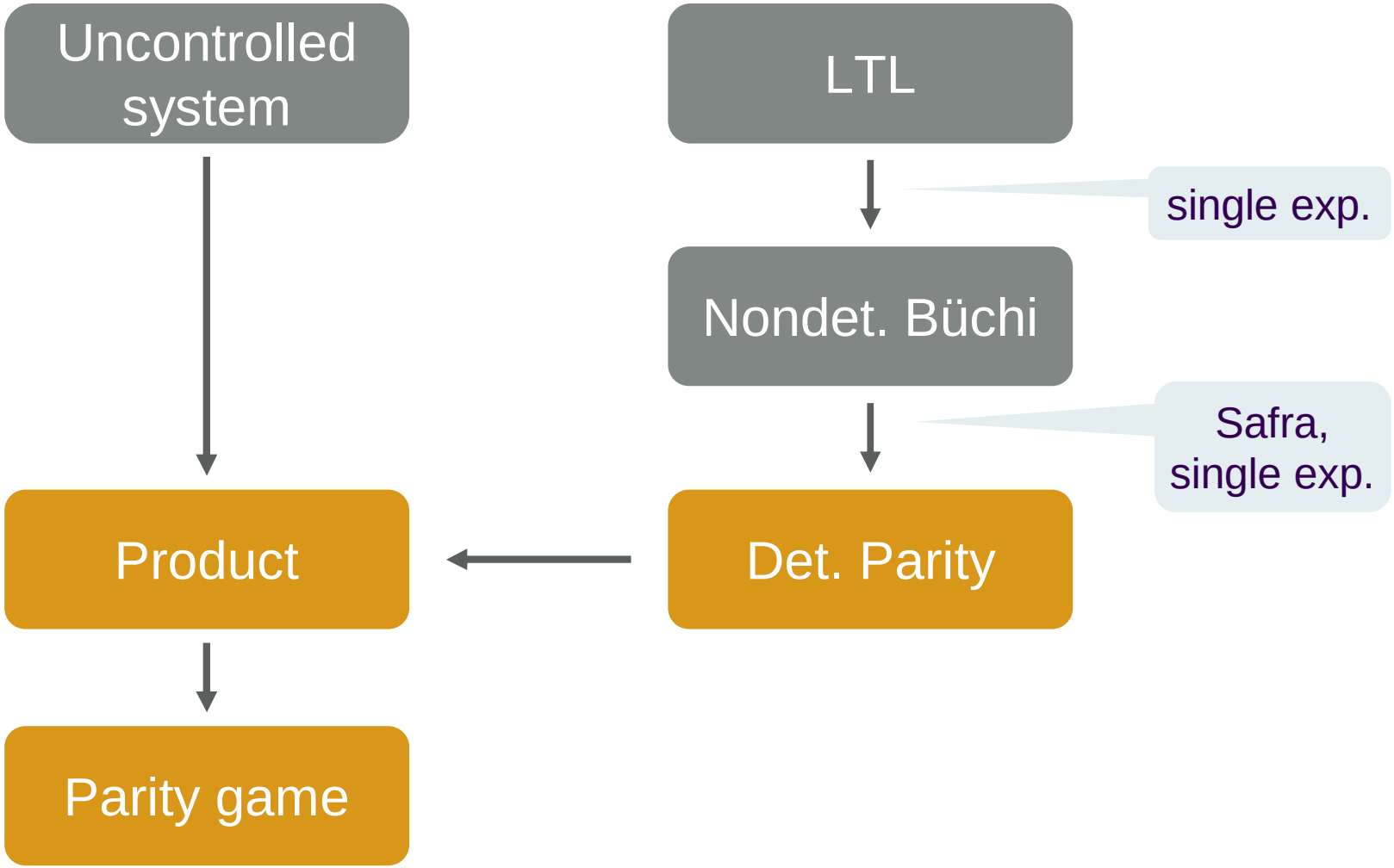
MODEL CHECKING RUNTIME
PNUELI-ZUCK MUTEX PROTOCOL

		#Clients	Our Imp. explicit	PRISM symb	PRISM+ Rabinizer symbolic	IscasMC explicit
(6)	$\mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}0 \vee \mathbf{FG}p2{\neq}0) \\ \wedge(\mathbf{GF}p2{=}0 \vee \mathbf{FG}p3{\neq}0) \\ \wedge(\mathbf{GF}p3{=}0 \vee \mathbf{FG}p1{\neq}0)\end{smallmatrix}]$	4	**< 1**	78	9	3
		5	**10**	1293	137	29
(7)	$\mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}0 \vee \mathbf{FG}p1{\neq}0) \\ \wedge(\mathbf{GF}p2{=}0 \vee \mathbf{FG}p2{\neq}0) \\ \wedge(\mathbf{GF}p3{=}0 \vee \mathbf{FG}p3{\neq}0)\end{smallmatrix}]$	4	**< 1**	**< 1**	61	2
		5	1	**< 1**	1077	27
(8)	$\mathbb{P}_{min=?}[\begin{smallmatrix}(\mathbf{GF}p1{\neq}10 \vee \mathbf{GF}p1{=}0 \vee \mathbf{FG}p1{=}1) \\ \wedge \mathbf{GF}p1{\neq}0 \wedge \mathbf{GF}p1{=}1\end{smallmatrix}]$	4	**< 1**	8	8	1
		5	**1**	145	190	16
(9)	$\mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{G}p1{\neq}10 \vee \mathbf{G}p2{\neq}10 \vee \mathbf{G}p3{\neq}10) \\ \wedge(\mathbf{FG}p1{\neq}1 \vee \mathbf{GF}p2{=}1 \vee \mathbf{GF}p3{=}1) \\ \wedge(\mathbf{FG}p2{\neq}1) \vee \mathbf{GF}p1{=}1 \vee \mathbf{GF}p3{=}1)\end{smallmatrix}]$	4	**5**	-	1195	8
		5	**99**	-	-	125
(10)	$\mathbb{P}_{min=?}[\begin{smallmatrix}\mathbf{FG}p1{\neq}0 \vee \mathbf{FG}p2{\neq}0 \\ \vee \mathbf{GF}p3{=}0 \vee (\mathbf{FG}p1{\neq}10 \\ \wedge \mathbf{GF}p2{=}10 \wedge \mathbf{GF}p3{=}10)\end{smallmatrix}]$	4	**1**	728	33	79
		5	**24**	-	486	-
(11)	$\mathbb{P}_{min=?}[f_{0,0}] = \mathbb{P}_{min=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}10)\mathbf{U} \\ (p2{=}10)\end{smallmatrix}]$	4	**< 1**	17	40	2
		5	**11**	257	715	23
(12)	$\mathbb{P}_{max=?}[f_{0,4}] = \mathbb{P}_{max=?}[\begin{smallmatrix}(\mathbf{GF}p1{=}10)\mathbf{U} \\ (\mathbf{XXXX}p2{=}10)\end{smallmatrix}]$	4	**< 1**	3	**< 1**	1
		5	5	20	**2**	20

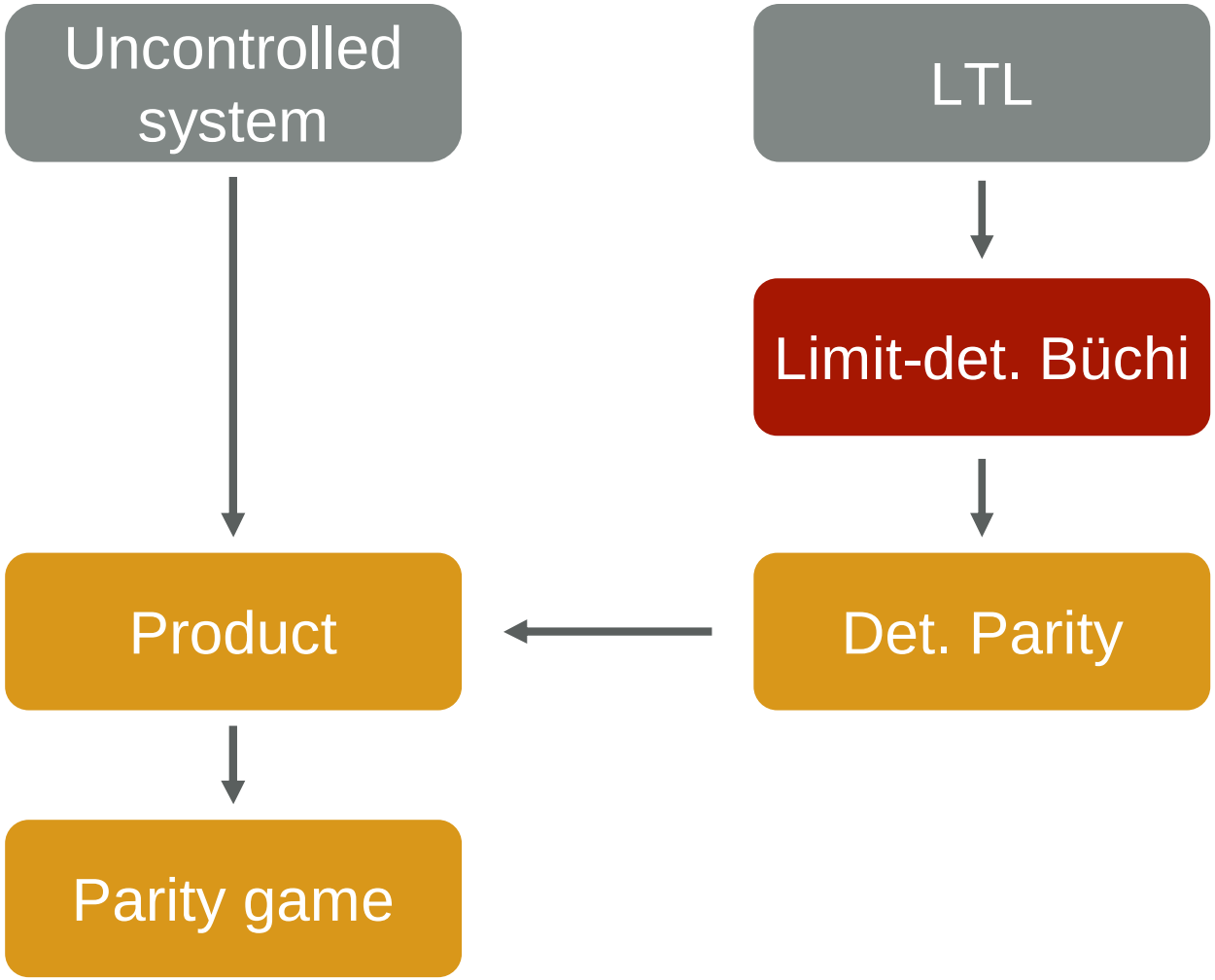
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS ?



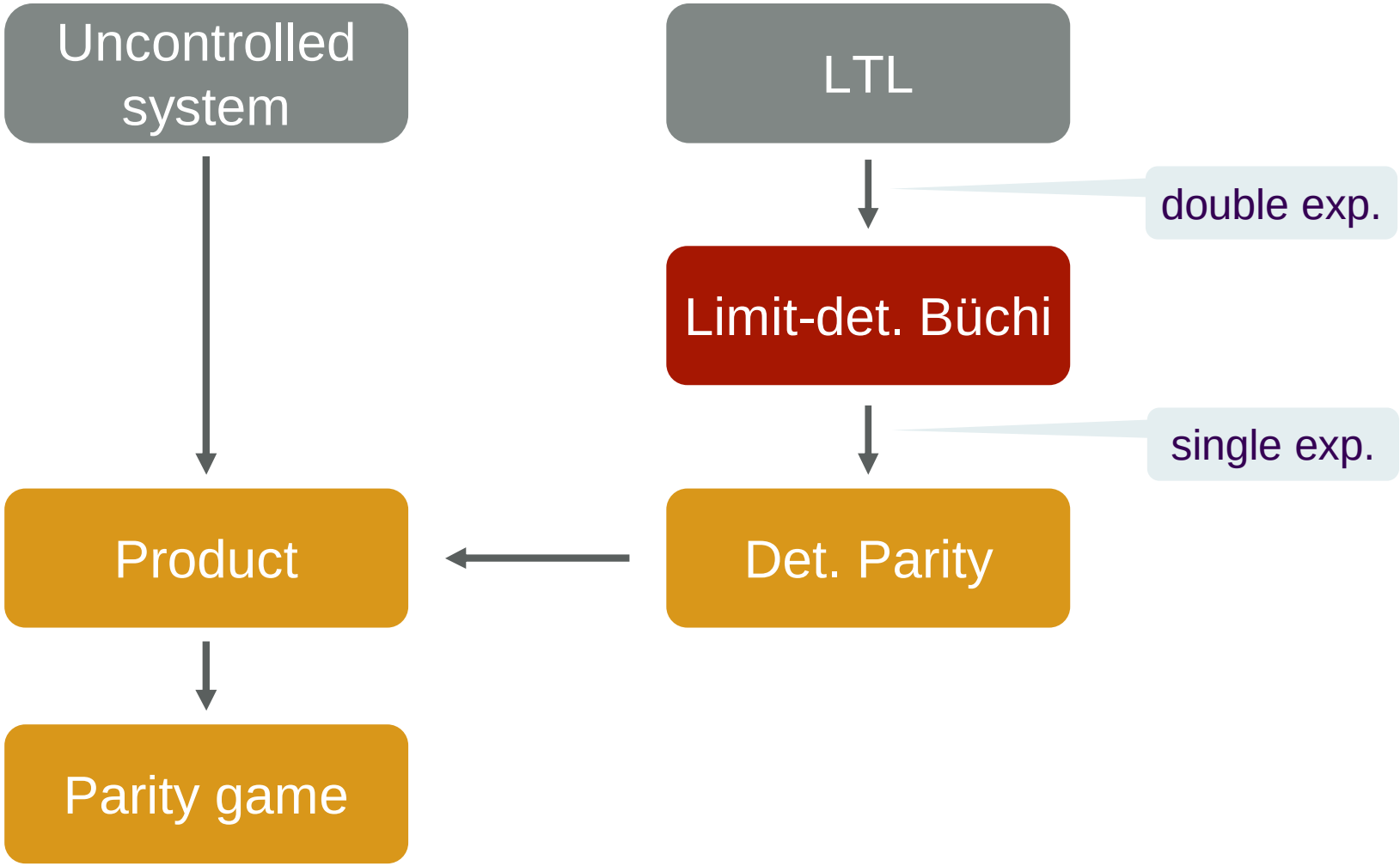
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS ?



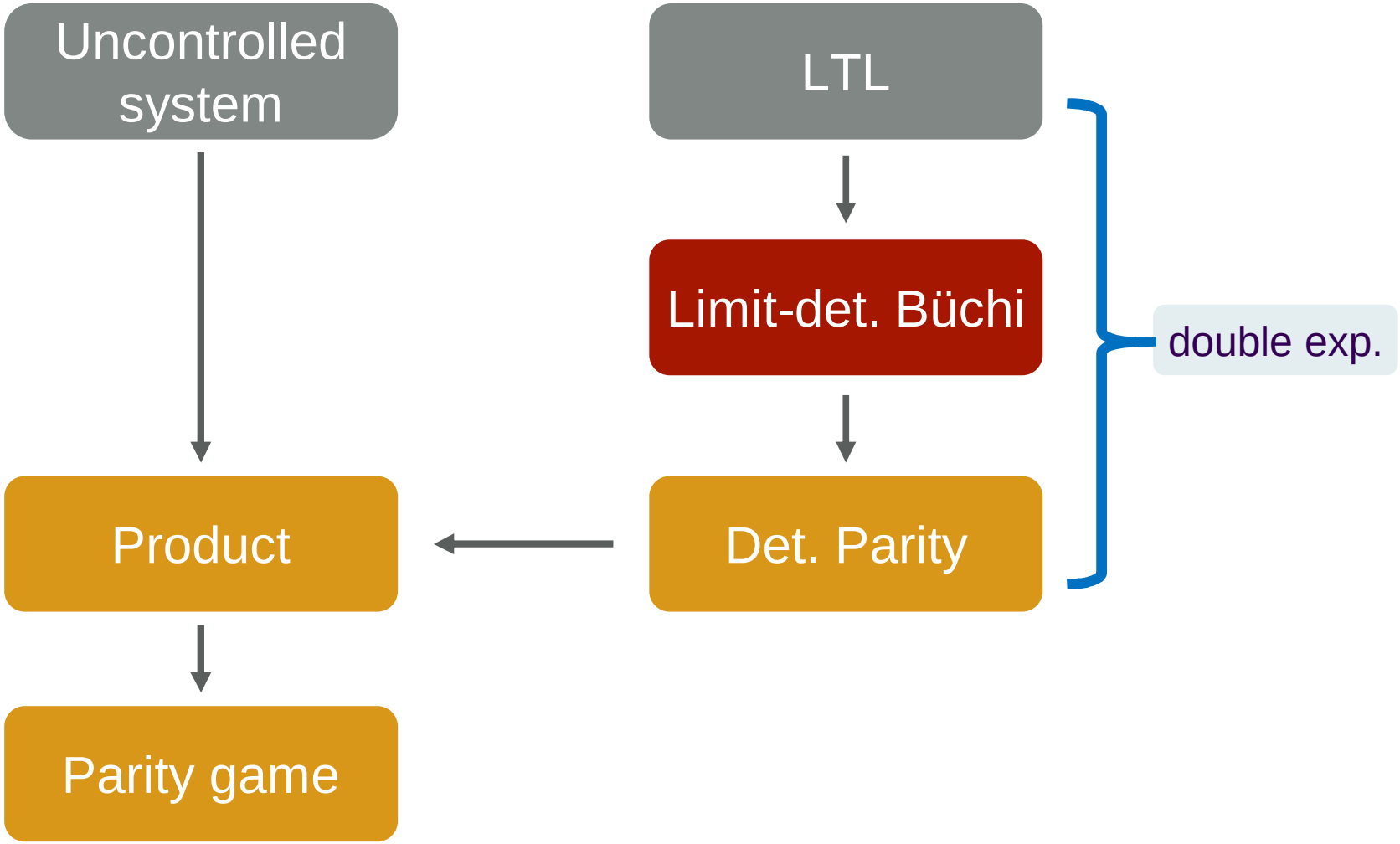
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS ?



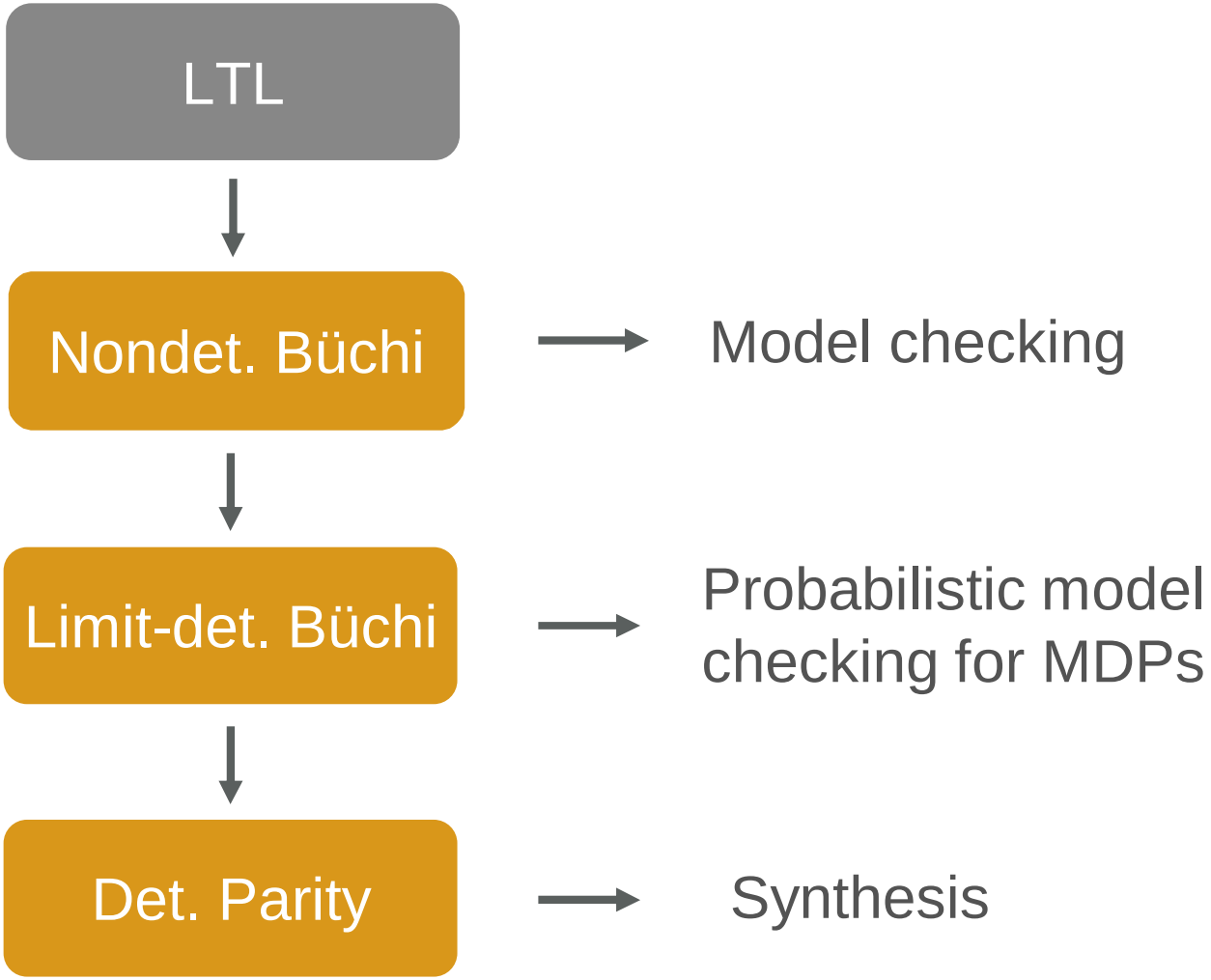
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS ?



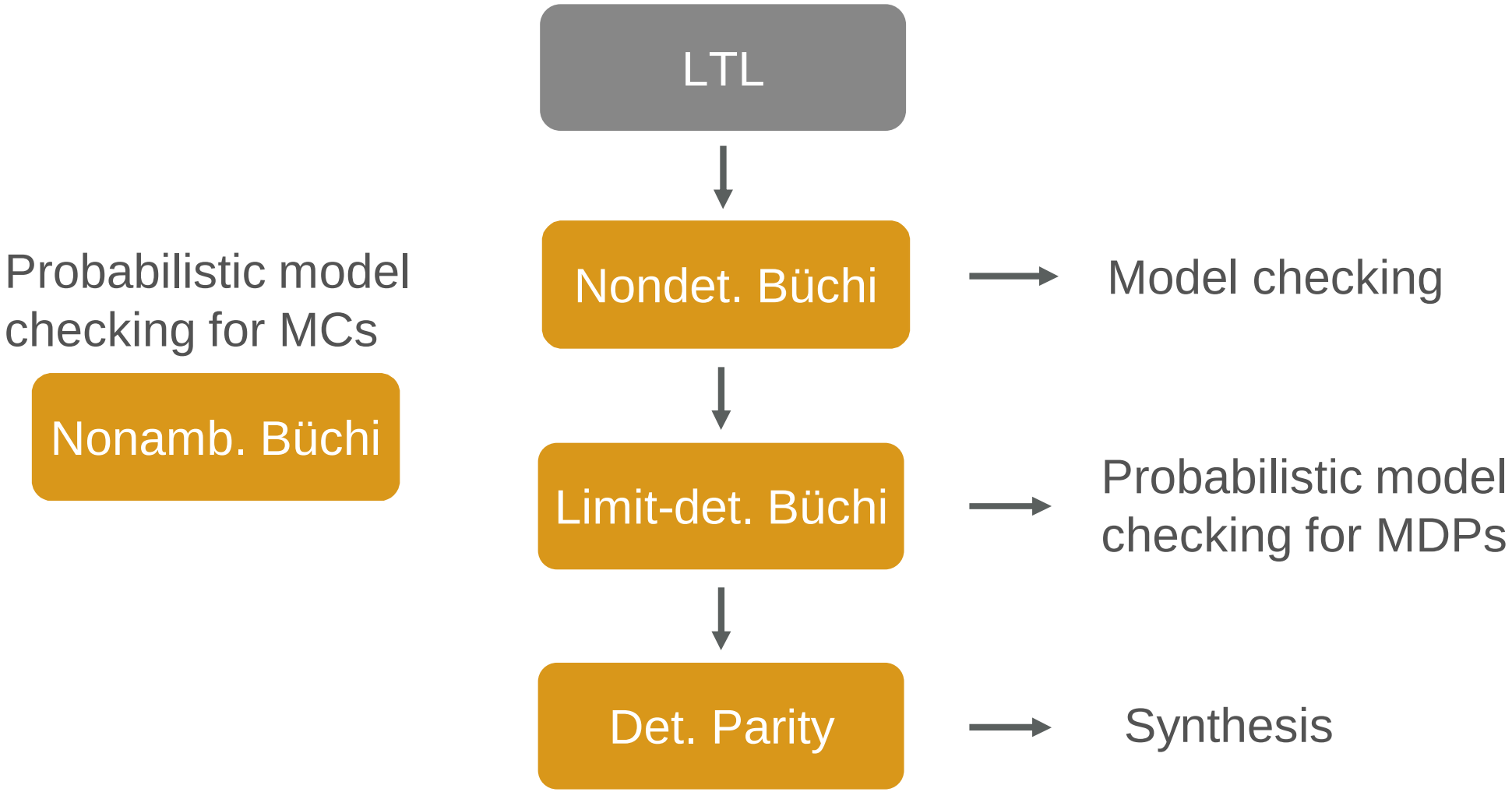
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS ?



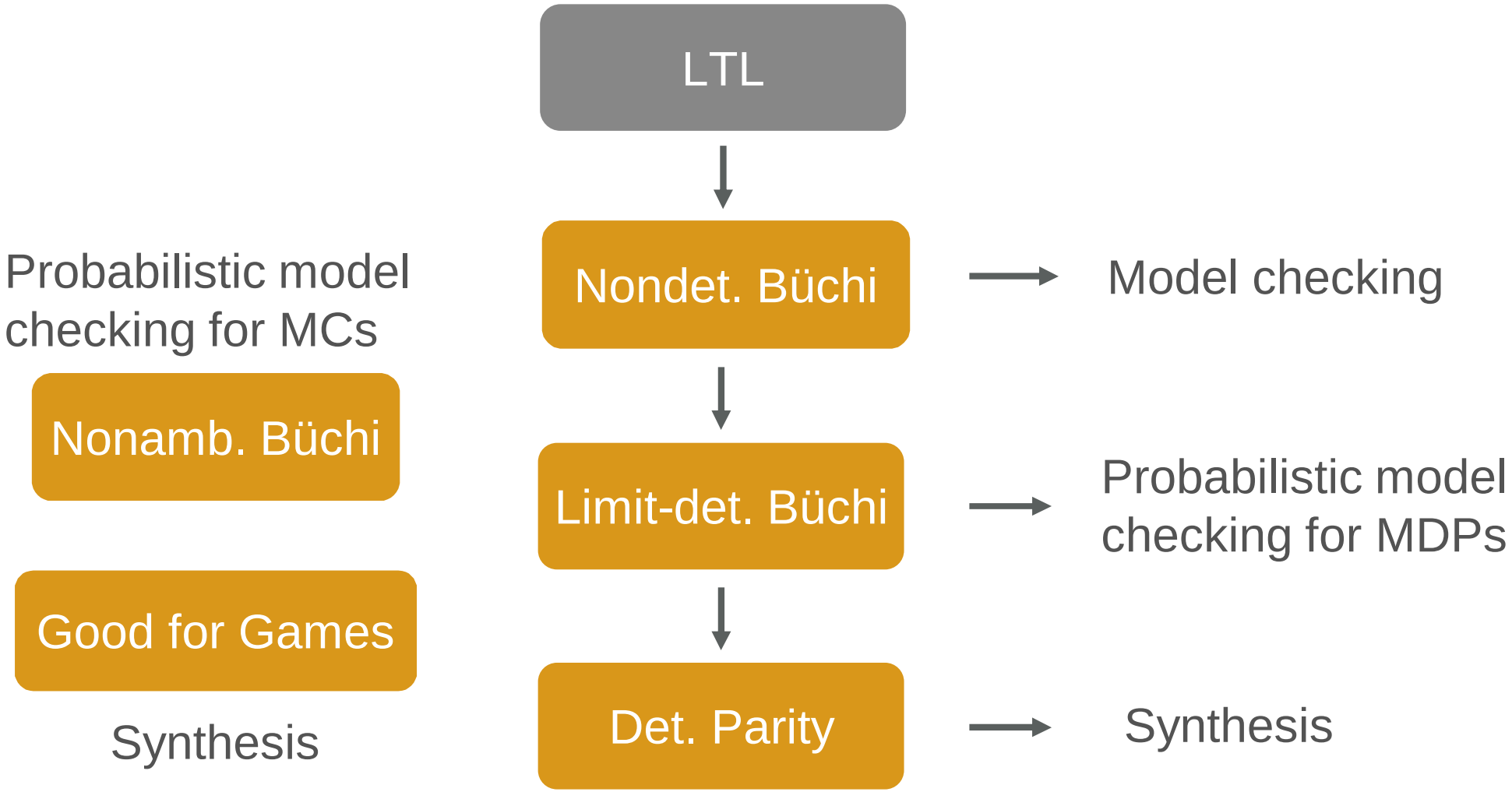
THE NEW PICTURE



THE NEW PICTURE



THE NEW PICTURE



CONCLUSION

- We have presented a translation from LTL to LDBA that
  - uses formulas as states
  - is modular
    - optimizations of any module helps to reduce state space!
  - yields in practice small ω-automata
  - is usable for quantitative prob. model checking without changing the algorithm.
  - can be also used as intermediate step to synthesis.
- Website: https://www7.in.tum.de/~sickert/projects/ltl2ldba/