

Model Checking Branching-Time Logics

Ph. Schnoebelen

<http://www.lsv-ens-cachan.fr/~phs>

Lab. Specification & Verification (LSV)

ENS de Cachan & CNRS

Cachan, France

Contents

We present new techniques and results on model checking for branching-time logics.

That is, logics like CTL or CTL^* , or rather **logics in-between CTL and CTL^*** .

Based on [Laroussinie, Markey, S., FoSSaCS'2001] and [S., ICALP'2003], available at <http://www.lsv.ens-cachan.fr/~phs>

(This work is part of a more general research program on algorithms and complexity for model checking temporal logics.)

Outline of the Talk

- From linear-time to branching-time logics
- Model-checking branching-time logics with oracles
- Some branching-time logics with a model-checking problem in Δ_2^p
- Completing the picture
- Conclusions and perspectives

Classics from branching-time model checking

Branching vs. linear time

In linear-time temporal logics, you consider what happens **along a run** of the system.

In branching-time temporal logics, you consider what happens **along the execution tree** of the system.

The runs are the **branches** of the execution tree.

Most famous examples:

$$\begin{aligned} LTL &= L(U, X) && \text{a linear-time logic} \\ CTL &= B(U, X) && \text{a branching-time logic} \\ ML &= B(X) \\ CTL^* &= B(LTL) = B(L(U, X)) = B^*(U, X) \end{aligned}$$

Branching-time logics are in principle more expressive than linear-time logics.

Some branching-time logics, e.g. *CTL*, have low model-checking complexity but they are not very expressive.

Classical question: what is the best compromise “**expressive power vs. complexity of model checking**”?

From linear to branching-time logics

From a (future) linear-time logic L , one derives its **branching-time extension** $B(L)$:

- for any $\varphi \in L$, allow $A\varphi$ as a new state formula,
- apply inductively.

NB. A and its dual E are **path quantifiers**.

Allows writing formulae like $AG[(X(EF^{\infty} \text{restart})UX \text{stop})]$.

Example: CTL^* is $B(LTL)$.

From linear to branching-time logics

From a (future) linear-time logic L , one derives its **branching-time extension** $B(L)$:

- for any $\varphi \in L$, allow $A\varphi$ as a new state formula,
- apply inductively.

NB. A and its dual E are **path quantifiers**.

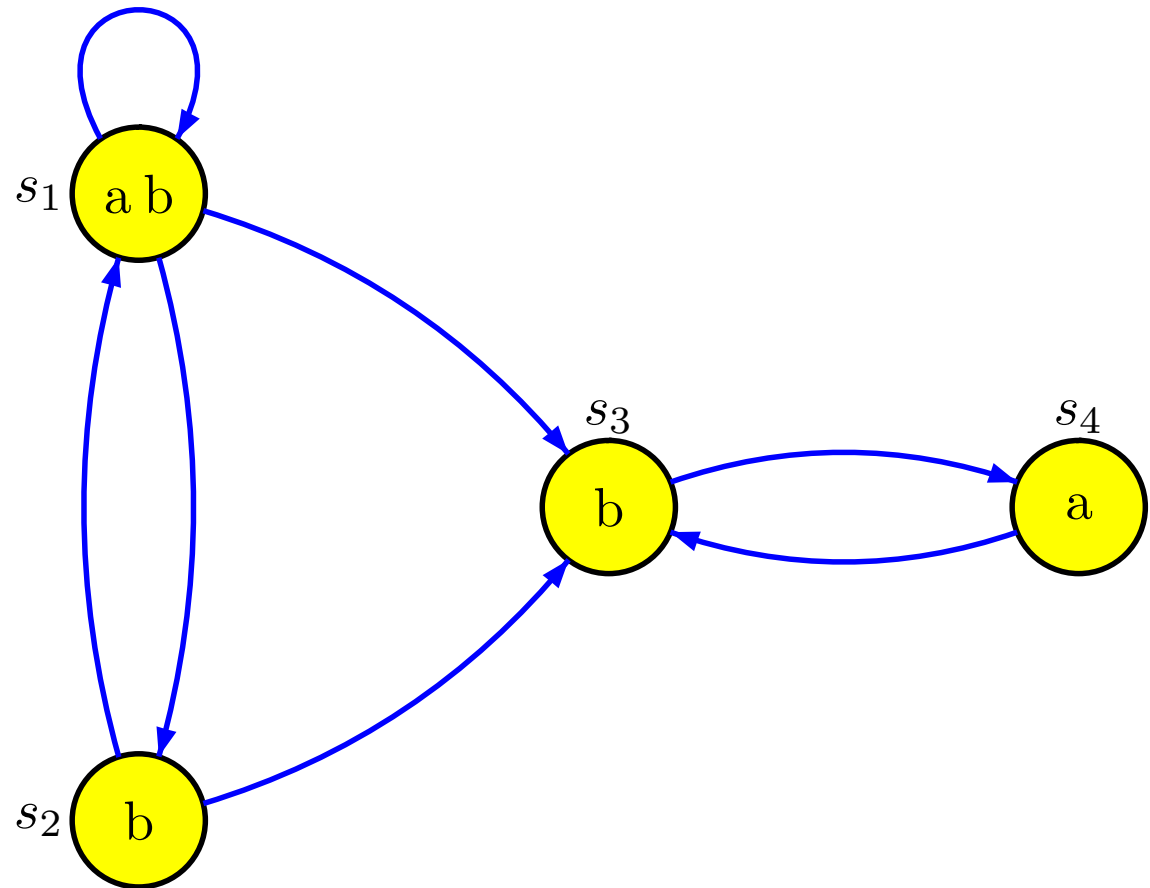
Allows writing formulae like $AG[(X(EF^{\infty}\text{ restart})UX\text{ stop})]$.

Example: CTL^* is $B(LTL)$.

Fact. From a model checking algorithm for L , one can derive a model checking algorithm for $B(L)$.

Model checking $B(L)$ [Emerson & Lei, 1987]

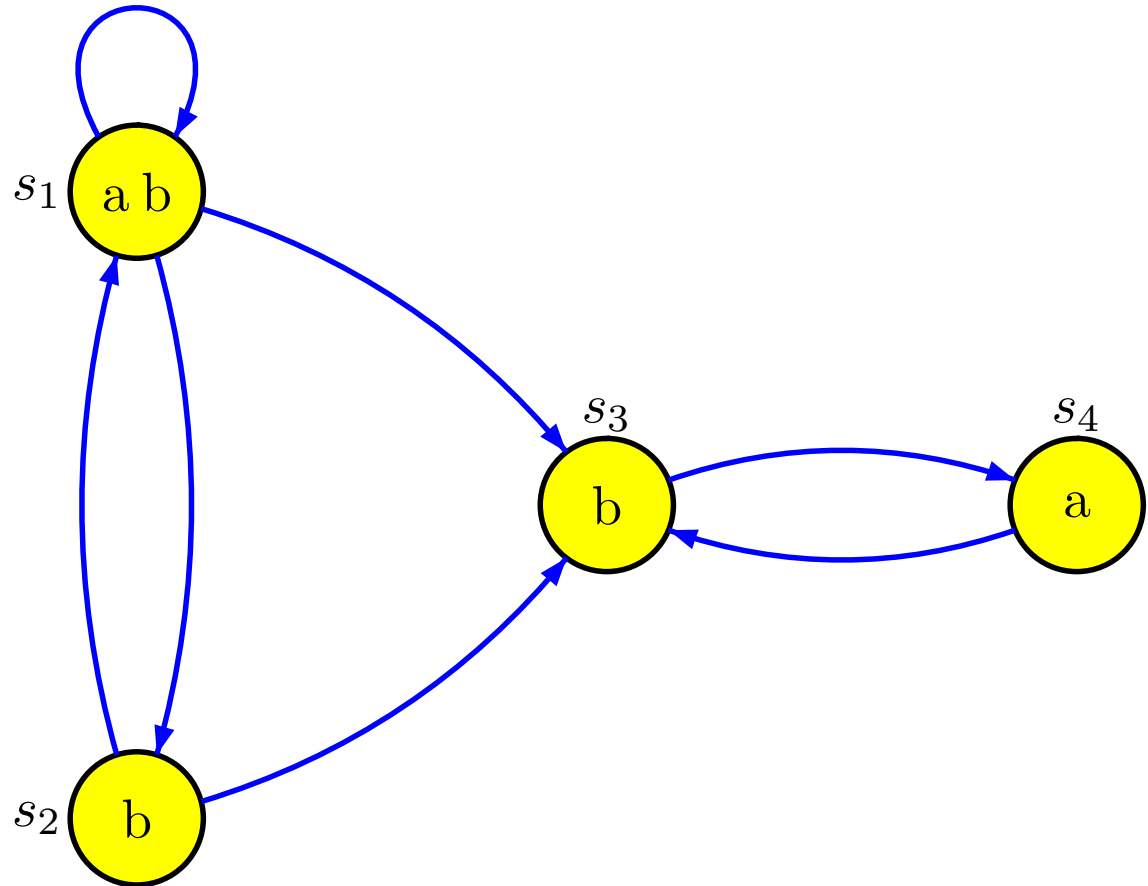
$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$



Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$



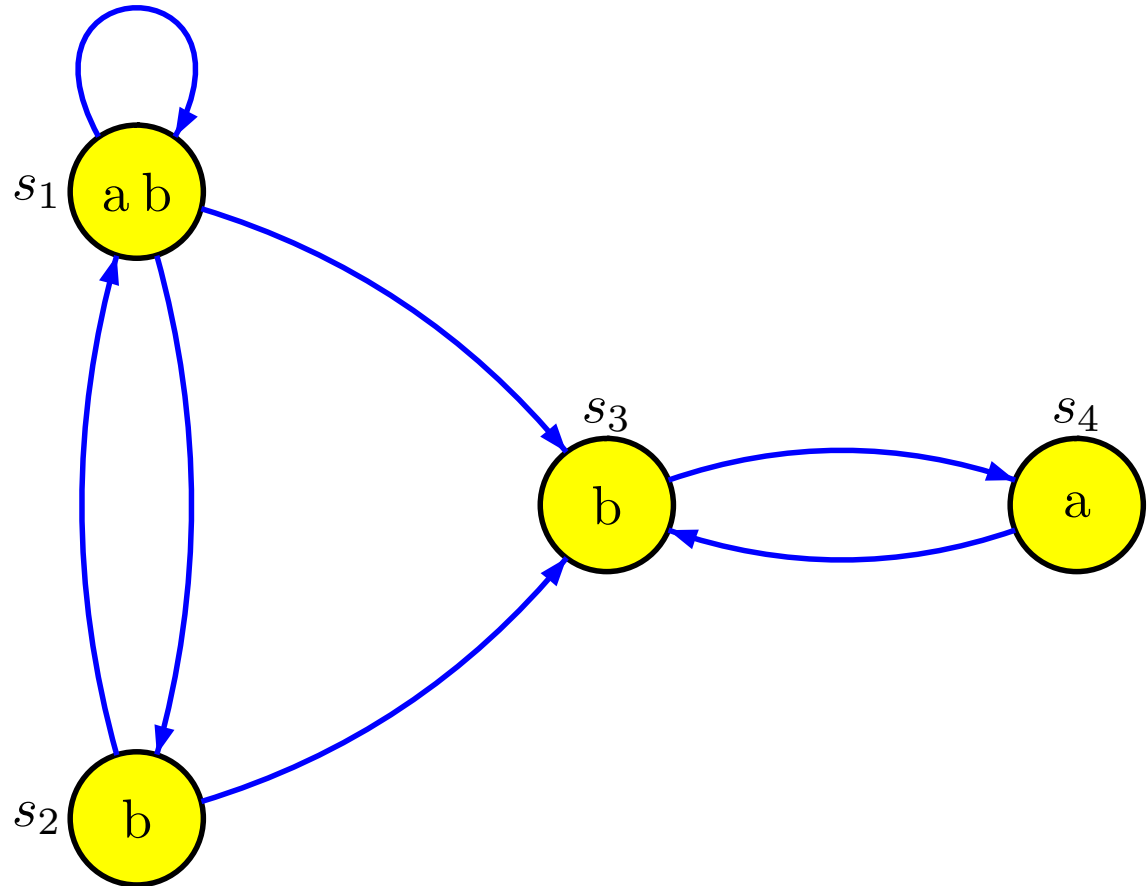
Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$



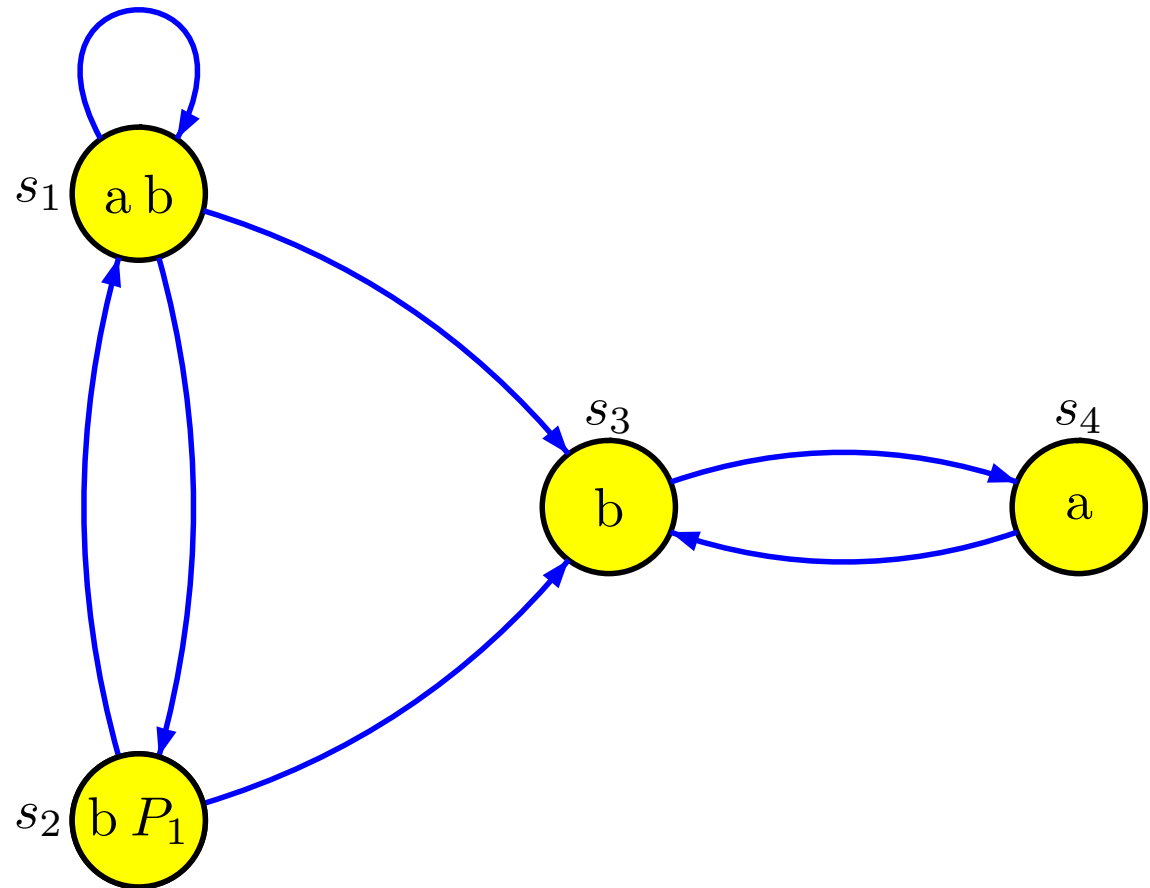
Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$



Model checking $B(L)$ [Emerson & Lei, 1987]

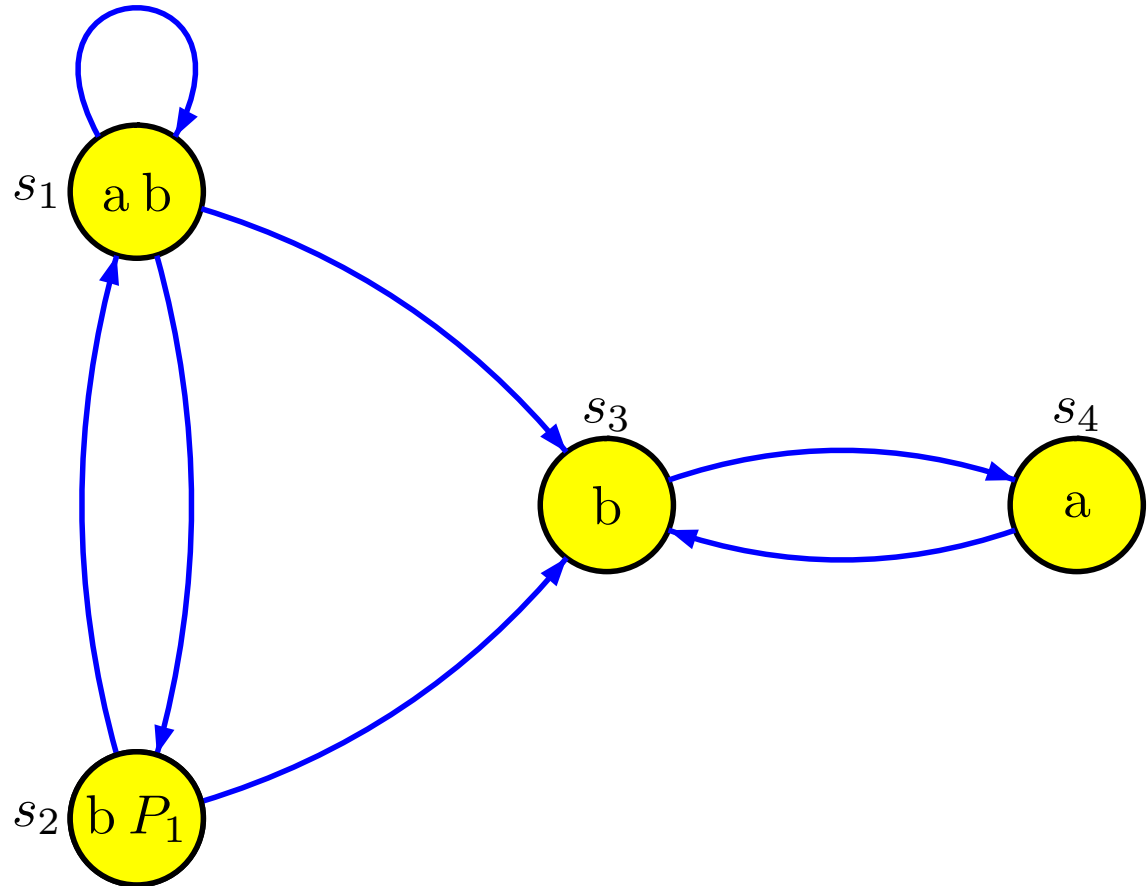
$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$

$$\varphi_2 = \overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg P_1$$



Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

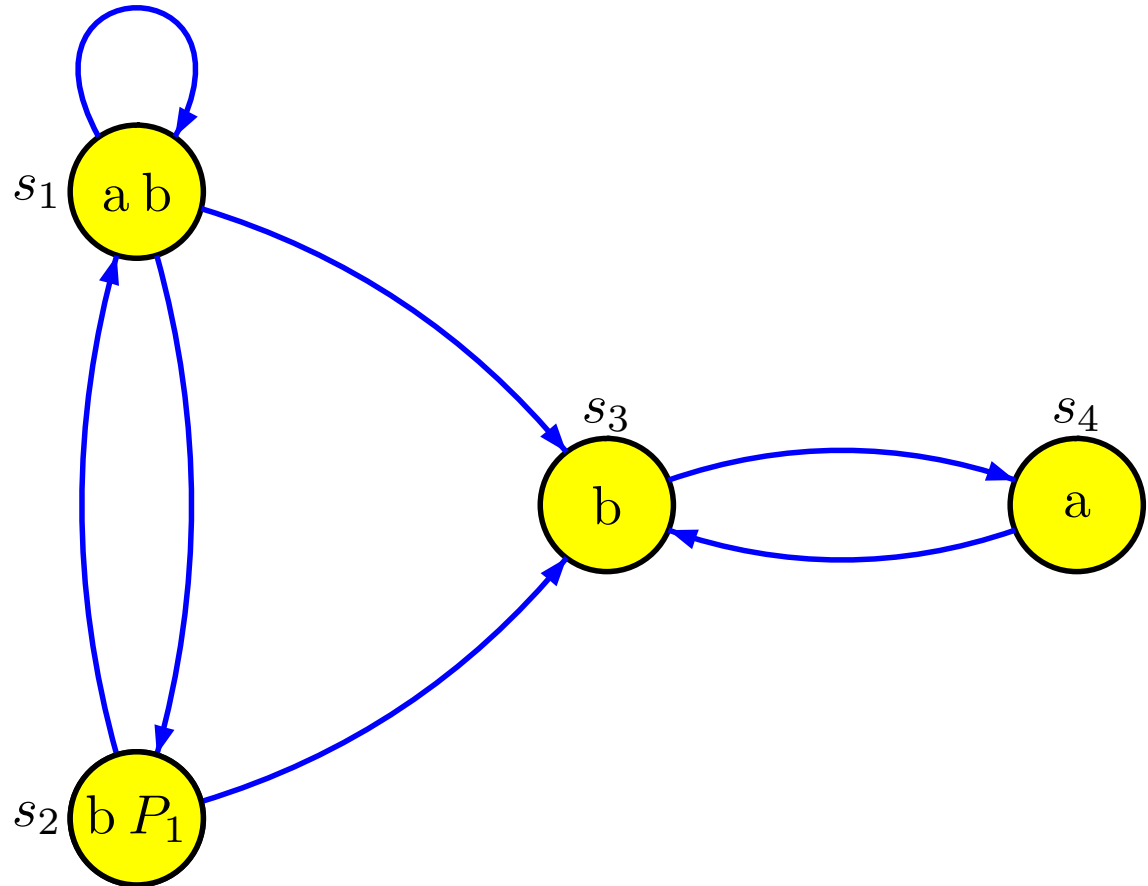
$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$

$$\varphi_2 = \overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg P_1$$

$$s_3, s_4 \models A \varphi_2$$

$$s_1, s_2 \not\models A \varphi_2$$



Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

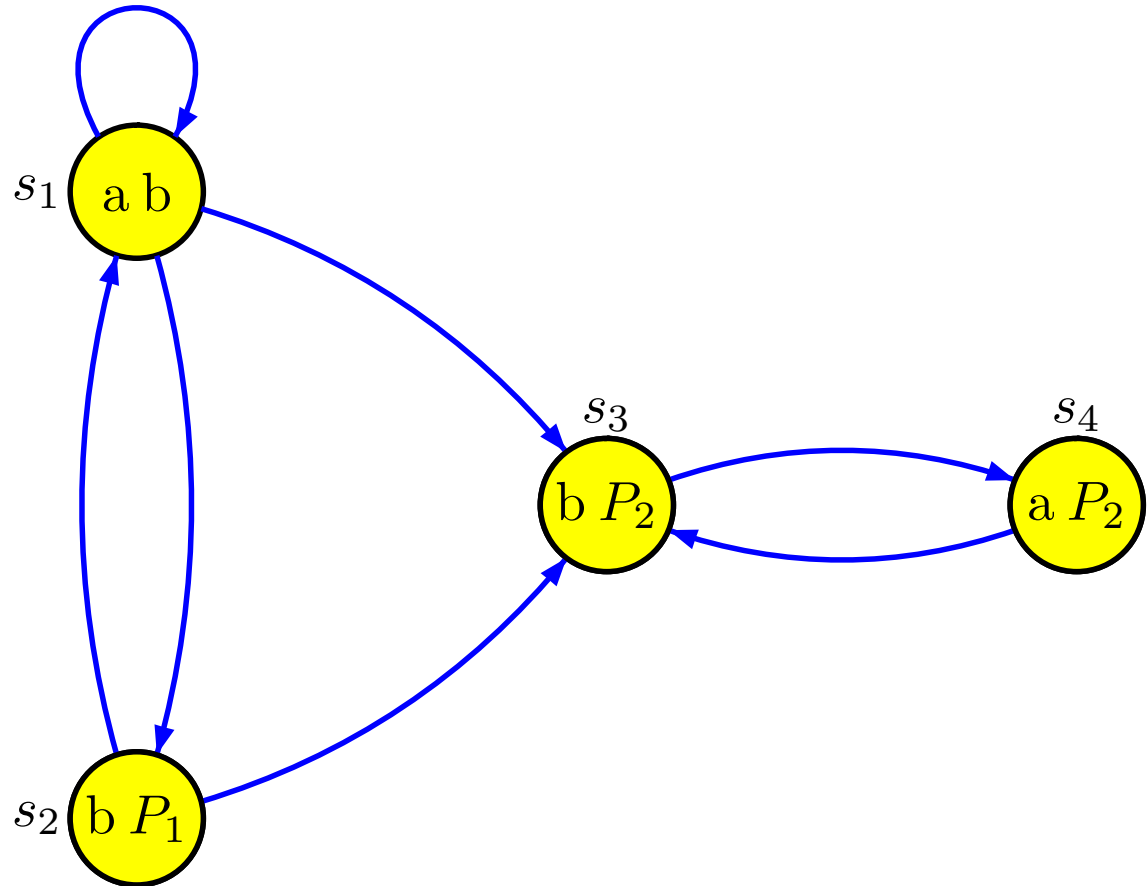
$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$

$$\varphi_2 = \overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg P_1$$

$$s_3, s_4 \models A \varphi_2$$

$$s_1, s_2 \not\models A \varphi_2$$



Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

$$s_2 \models A \varphi_1$$

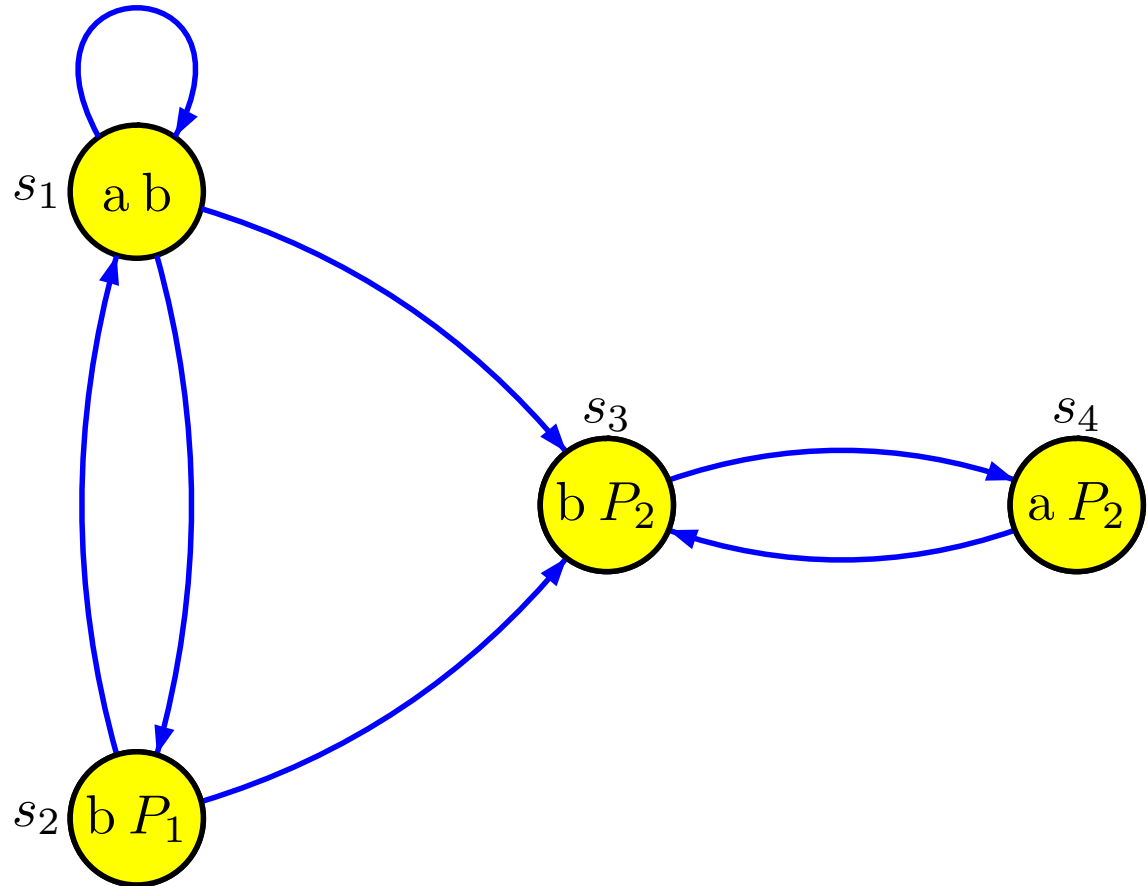
$$s_1, s_3, s_4 \not\models A \varphi_1$$

$$\varphi_2 = \overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg P_1$$

$$s_3, s_4 \models A \varphi_2$$

$$s_1, s_2 \not\models A \varphi_2$$

$$\varphi_3 = X P_2$$



Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$

$$\varphi_2 = \overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg P_1$$

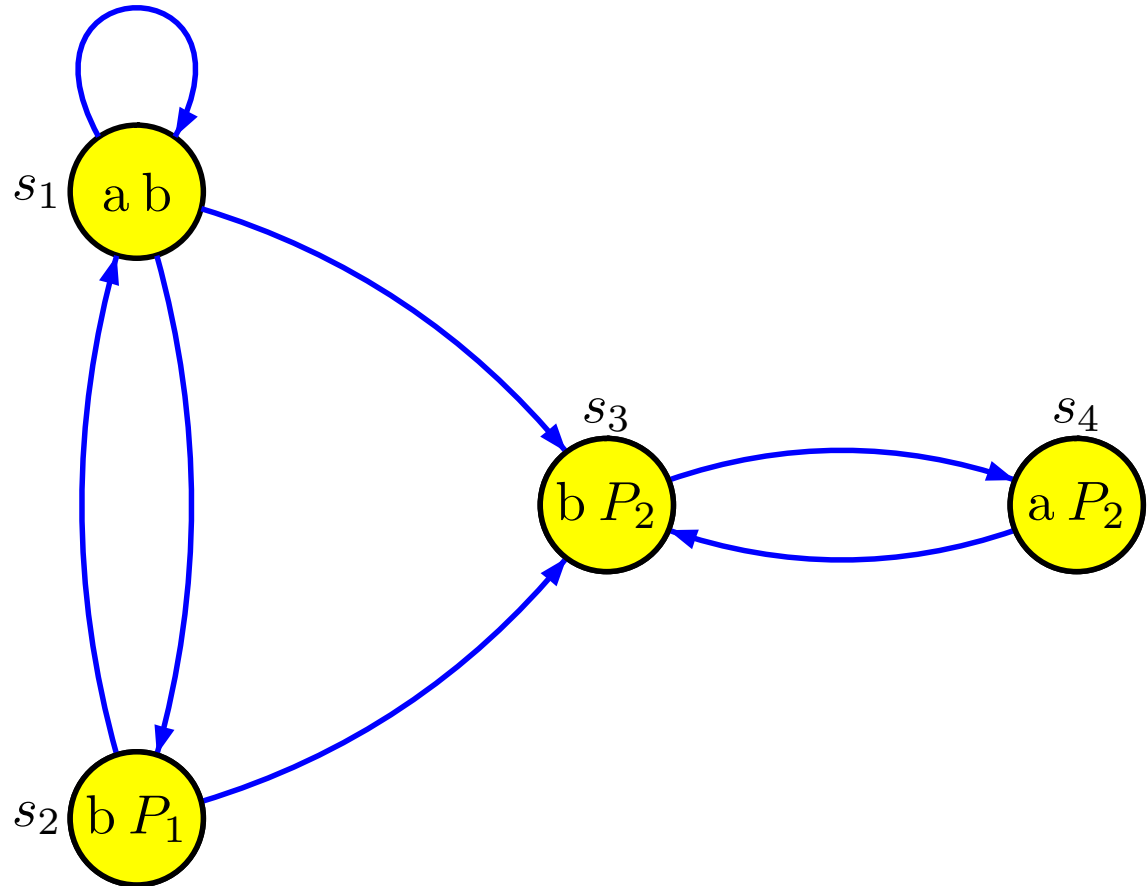
$$s_3, s_4 \models A \varphi_2$$

$$s_1, s_2 \not\models A \varphi_2$$

$$\varphi_3 = X P_2$$

$$s_3, s_4 \models A \varphi_3$$

$$s_1, s_2 \not\models A \varphi_3$$



Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg AF(Xb \wedge \neg a))$$

$$\varphi_1 = F(Xb \wedge \neg a)$$

$$s_2 \models A \varphi_1$$

$$s_1, s_3, s_4 \not\models A \varphi_1$$

$$\varphi_2 = \overset{\infty}{F}\neg a \wedge \overset{\infty}{G}\neg P_1$$

$$s_3, s_4 \models A \varphi_2$$

$$s_1, s_2 \not\models A \varphi_2$$

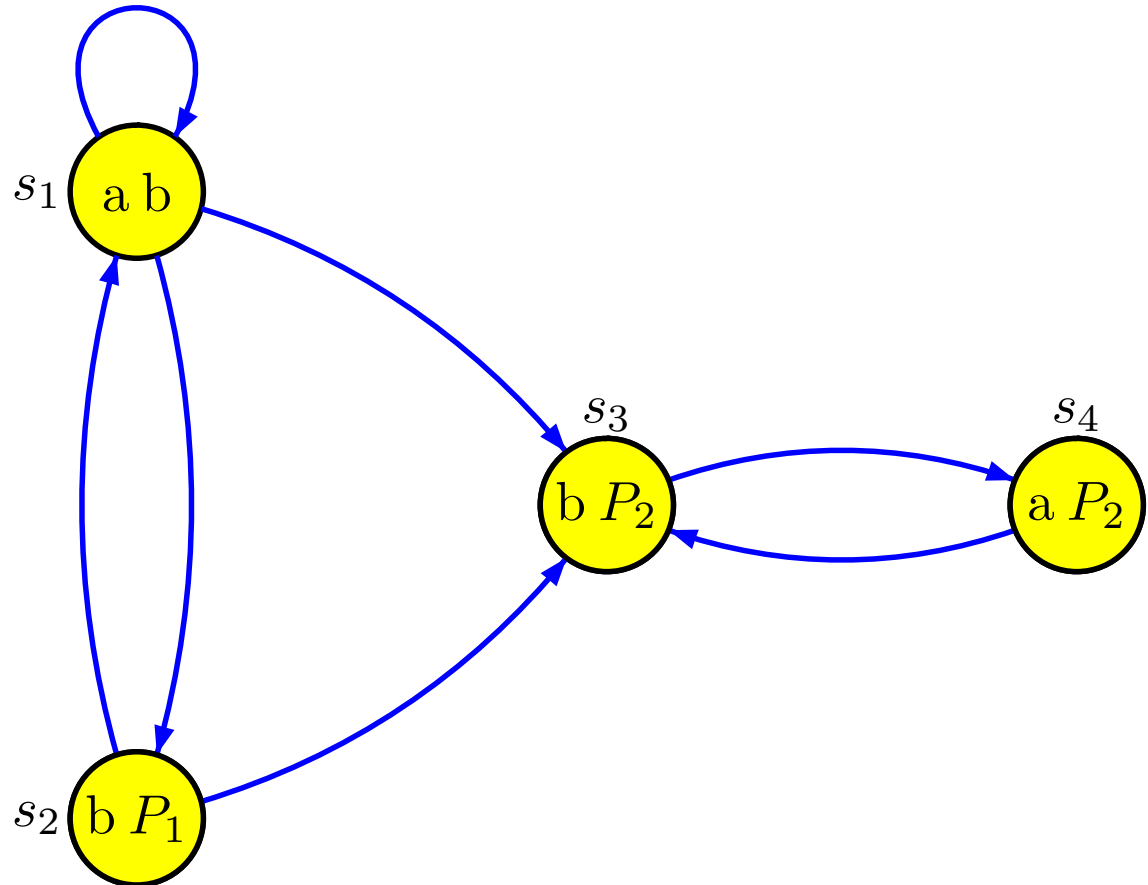
$$\varphi_3 = X P_2$$

$$s_3, s_4 \models A \varphi_3$$

$$s_1, s_2 \not\models A \varphi_3$$

$$s_3, s_4 \models \varphi$$

$$s_1, s_2 \not\models \varphi$$



Upper bounds

Fact. If model checking for L is in \mathcal{C} , then model checking for $B(L)$ is in $P^{\mathcal{C}}$.

NB. $P^{\mathcal{C}}$ is the class of problems solvable by a deterministic polynomial-time Turing machine that has access to an oracle in \mathcal{C} .

In fact, model checking $\varphi \in B(L)$ on some S can be done with $\text{nb_states}(S) \times \text{nb_path_quantifiers}(\varphi)$ invocations of the model checker for L .

Upper bounds

Fact. If model checking for L is in \mathcal{C} , then model checking for $B(L)$ is in $P^{\mathcal{C}}$.

NB. $P^{\mathcal{C}}$ is the class of problems solvable by a deterministic polynomial-time Turing machine that has access to an oracle in \mathcal{C} .

In fact, model checking $\varphi \in B(L)$ on some S can be done with $\text{nb_states}(S) \times \text{nb_path_quantifiers}(\varphi)$ invocations of the model checker for L .

Example 1. Model checking for LTL is in PSPACE, hence model checking for $B(LTL)$, a.k.a. CTL^* , is in P^{PSPACE} , that is, in PSPACE.

Upper bounds

Fact. If model checking for L is in \mathcal{C} , then model checking for $B(L)$ is in $P^{\mathcal{C}}$.

NB. $P^{\mathcal{C}}$ is the class of problems solvable by a deterministic polynomial-time Turing machine that has access to an oracle in \mathcal{C} .

In fact, model checking $\varphi \in B(L)$ on some S can be done with $\text{nb_states}(S) \times \text{nb_path_quantifiers}(\varphi)$ invocations of the model checker for L .

Example 1. Model checking for LTL is in PSPACE, hence model checking for $B(LTL)$, a.k.a. CTL^* , is in P^{PSPACE} , that is, in PSPACE.

Example 2. Model checking for $L_0 \stackrel{\text{def}}{=} \{X P, P U P'\}$ is in NL, hence model checking for $B(L_0)$, a.k.a. CTL , is in P^{NL} , that is, in P.

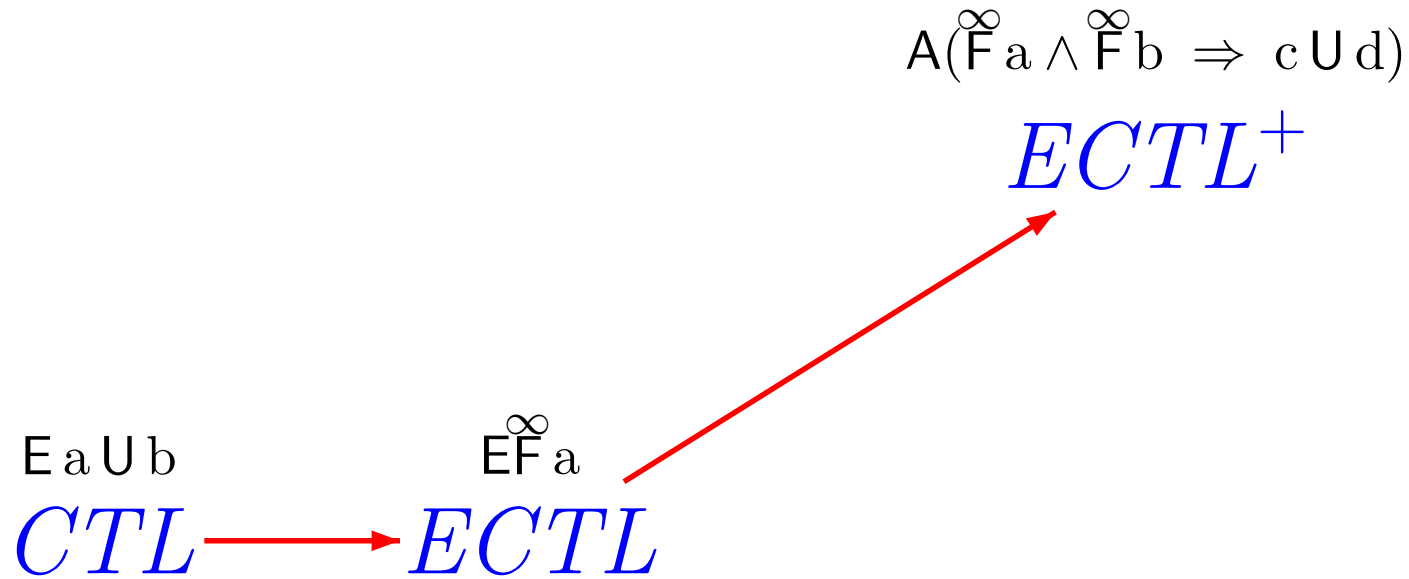
What expressivity/complexity compromise?

E a U b
CTL

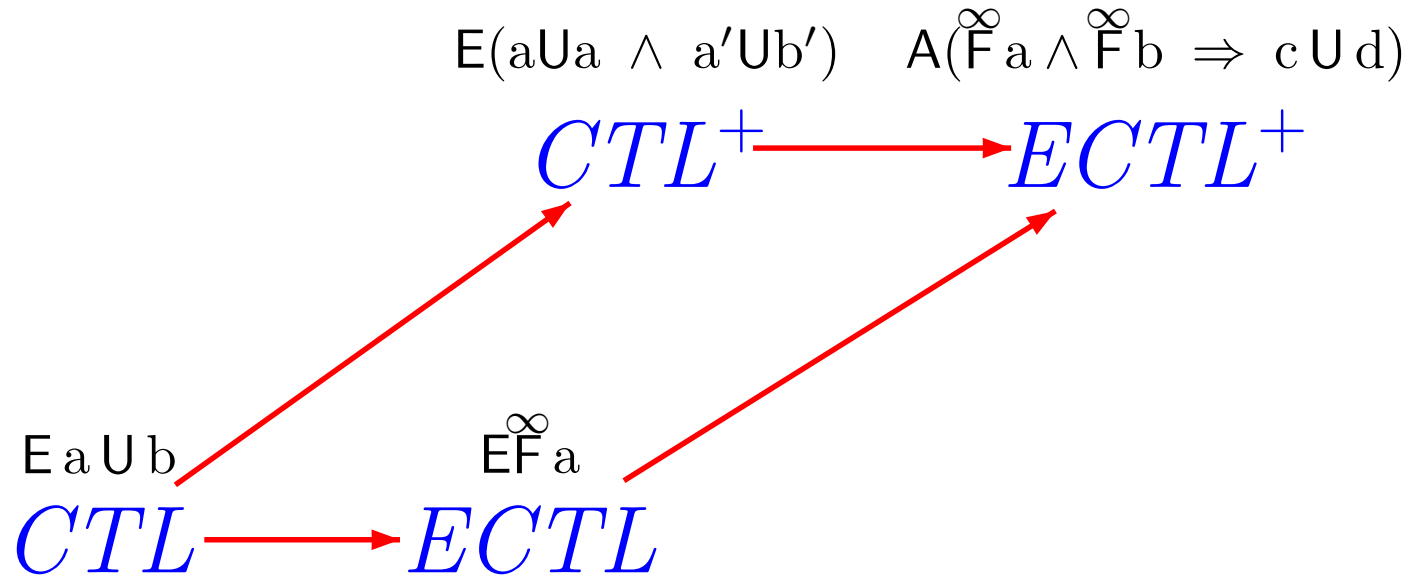
What expressivity/complexity compromise?

$E a U b$ $E F^{\infty} a$
CTL \longrightarrow *ECTL*

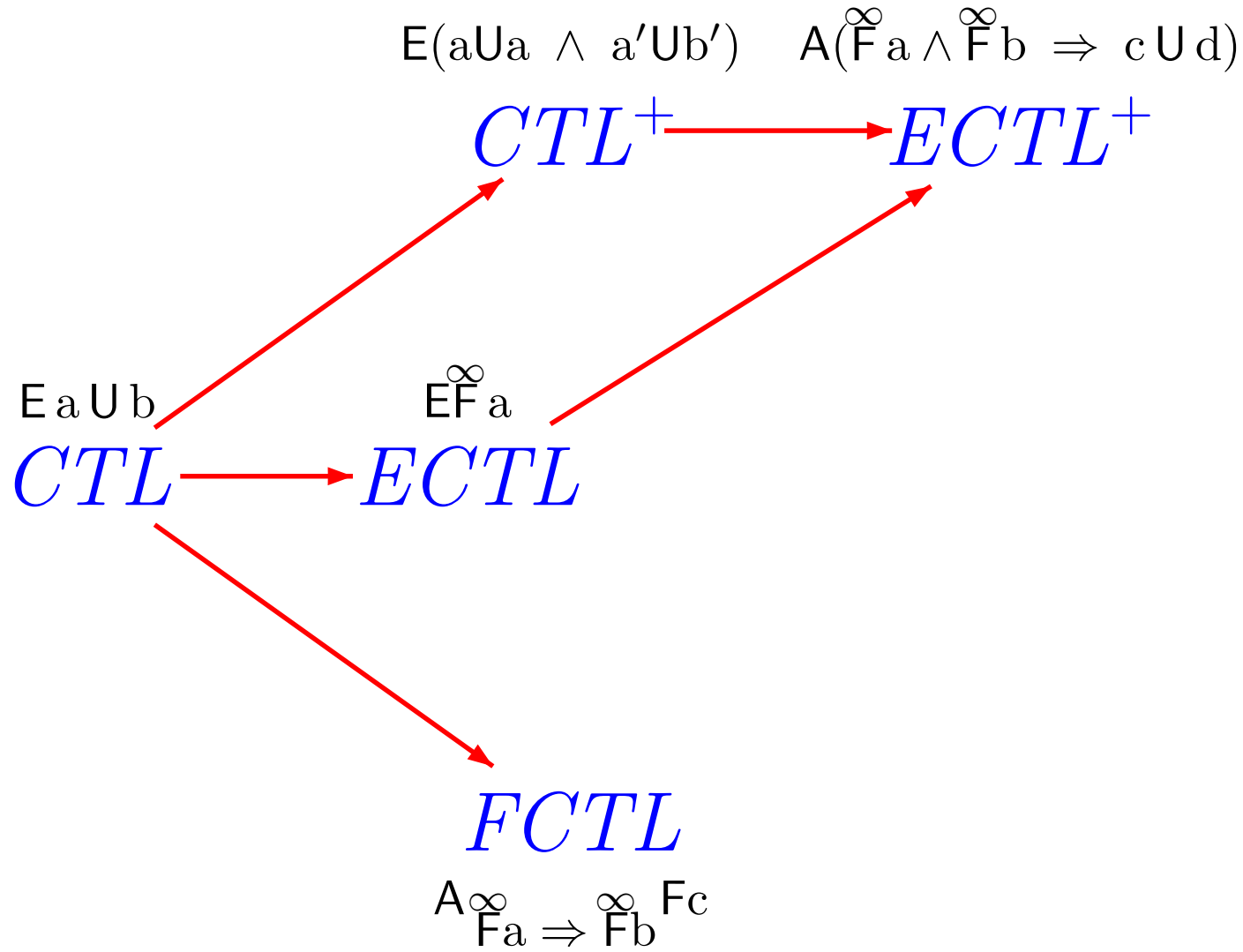
What expressivity/complexity compromise?



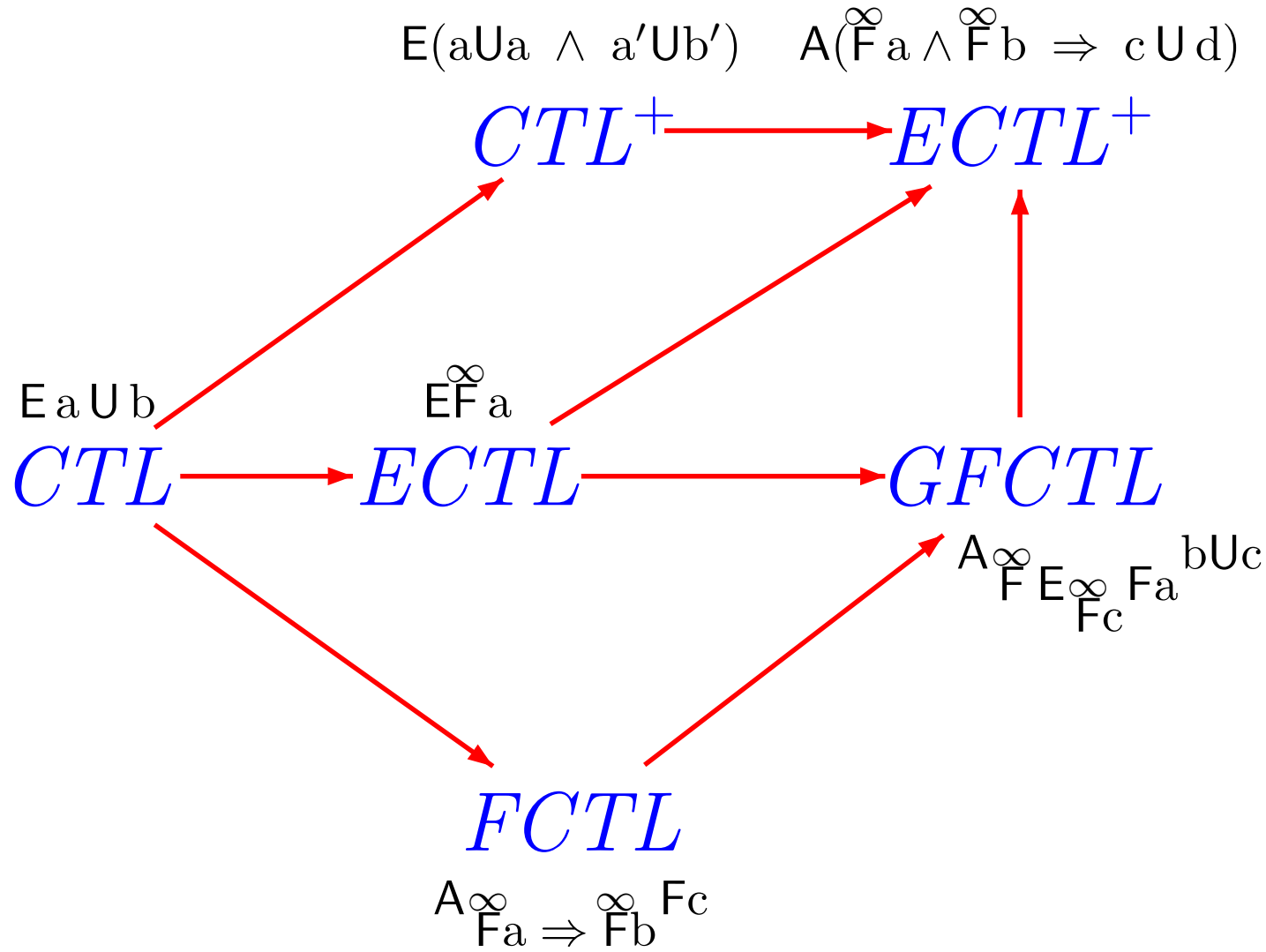
What expressivity/complexity compromise?



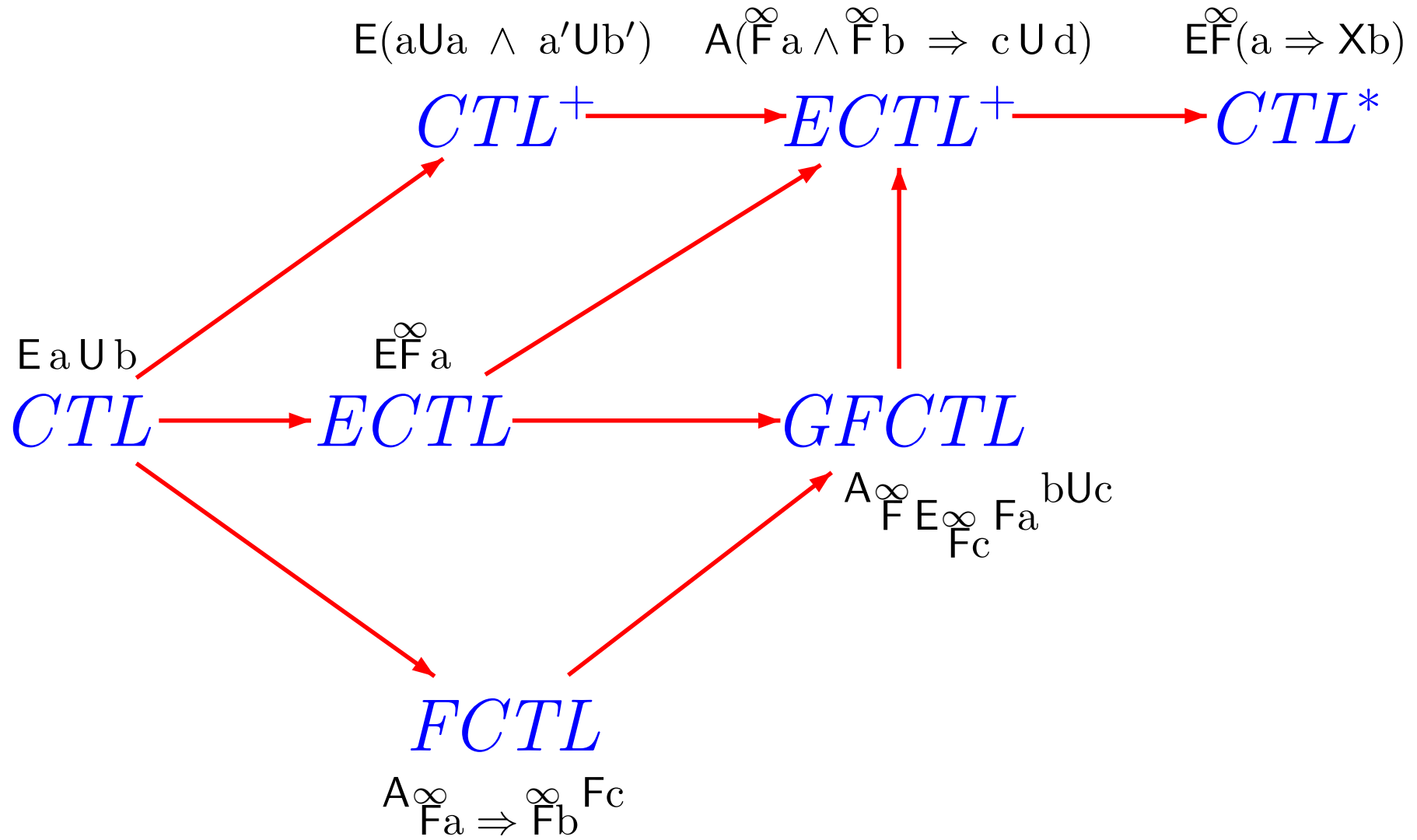
What expressivity/complexity compromise?



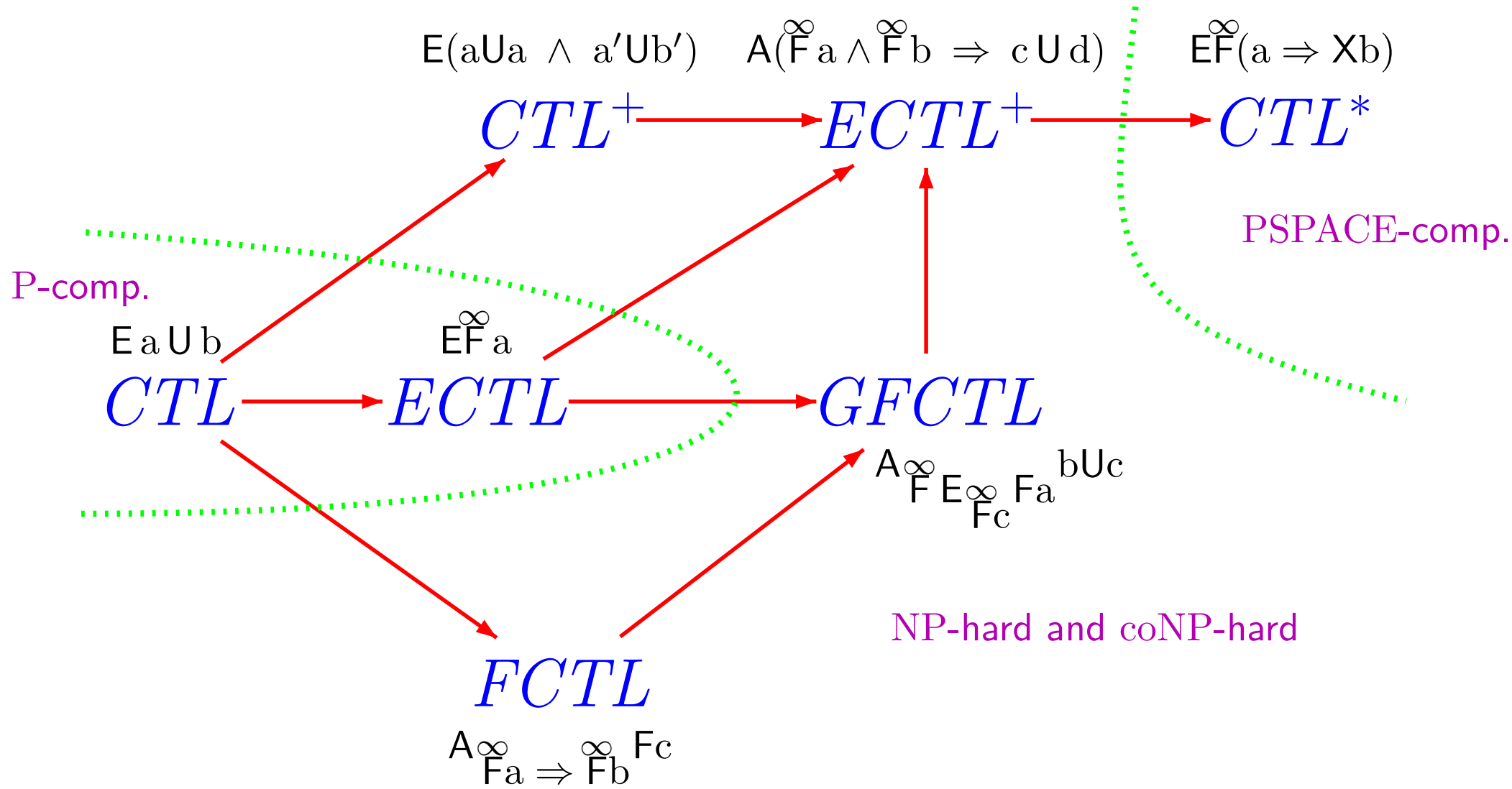
What expressivity/complexity compromise?



What expressivity/complexity compromise?



What expressivity/complexity compromise?

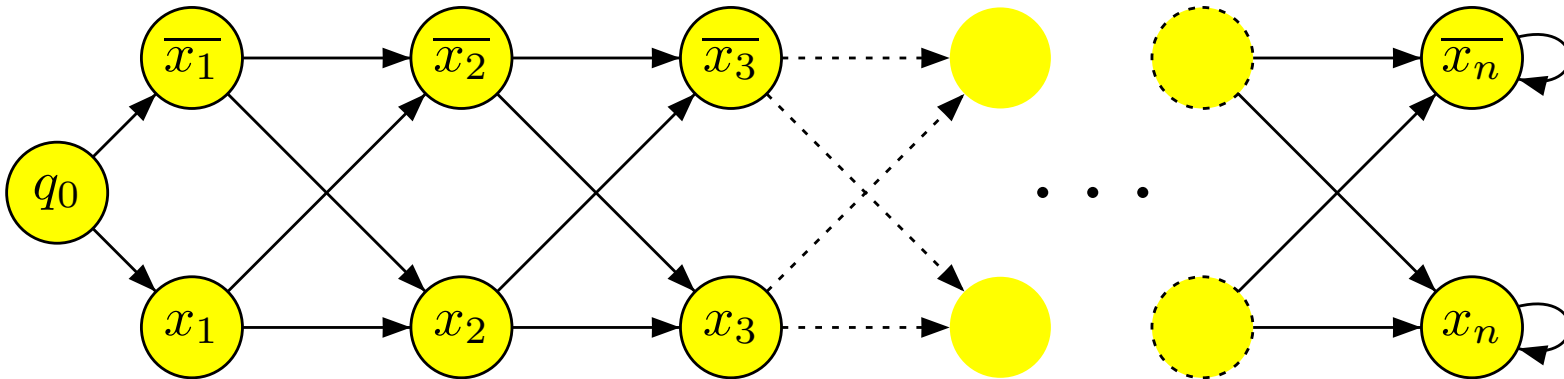


NP-complete path modalities

Let \mathcal{I} be a 3SAT instance. E.g. \mathcal{I} is $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \dots) \wedge \dots$ "

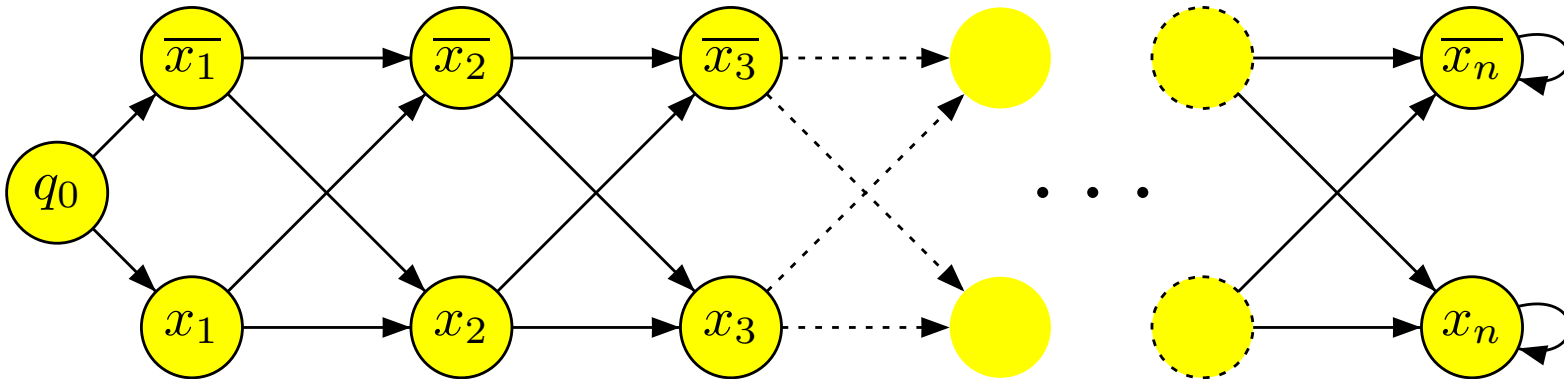
NP-complete path modalities

Let \mathcal{I} be a 3SAT instance. E.g. \mathcal{I} is $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \dots) \wedge \dots$



NP-complete path modalities

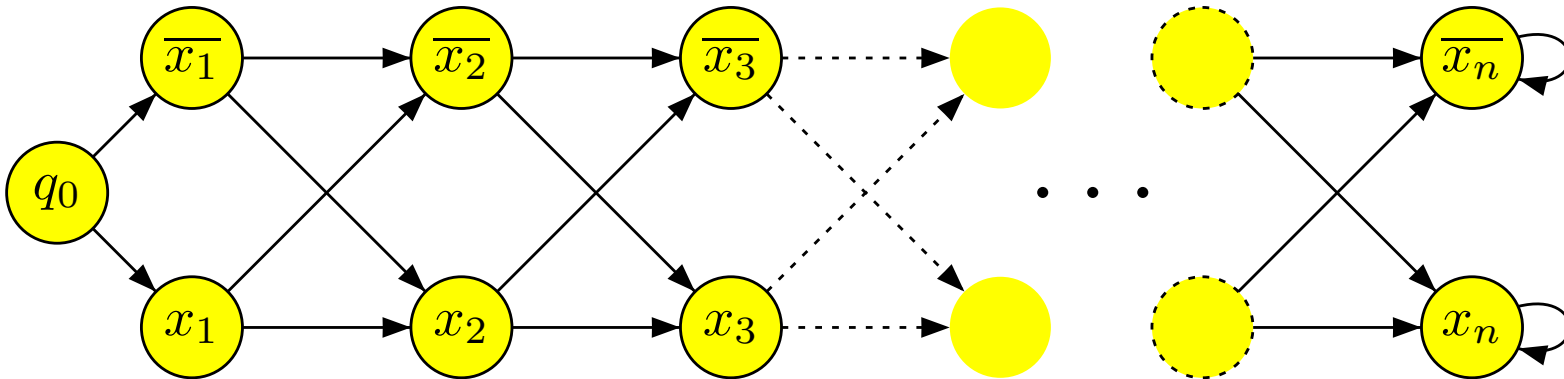
Let \mathcal{I} be a 3SAT instance. E.g. \mathcal{I} is $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \dots) \wedge \dots$



\mathcal{I} is satisfiable iff $q_0 \models E \left[(F x_1 \vee F \overline{x_2} \vee F \overline{x_4}) \wedge (F \overline{x_1} \vee \dots) \wedge \dots \right]$

NP-complete path modalities

Let \mathcal{I} be a 3SAT instance. E.g. \mathcal{I} is $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \dots) \wedge \dots$

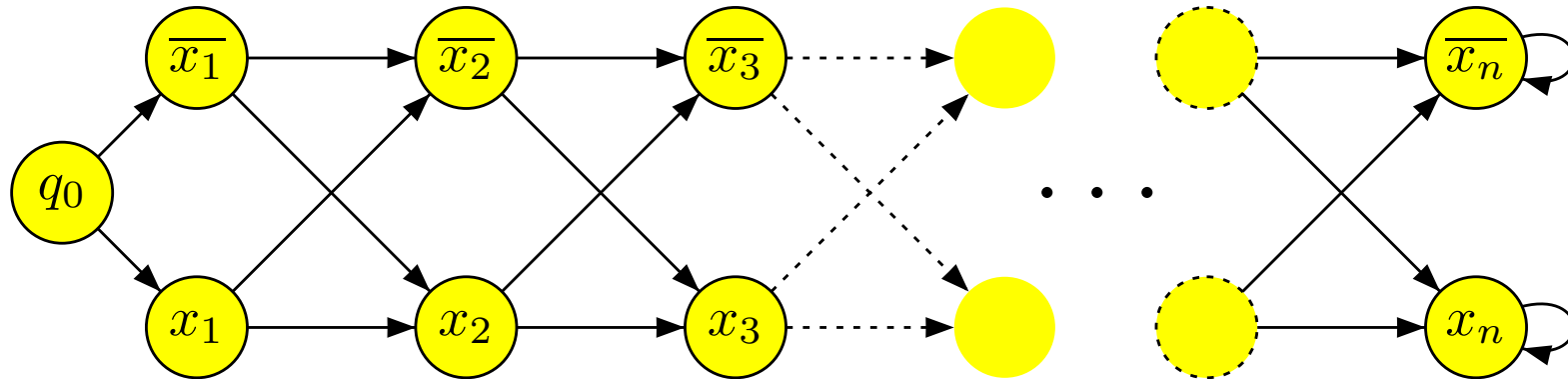


\mathcal{I} is satisfiable iff $q_0 \models E \left[(F x_1 \vee F \overline{x_2} \vee F \overline{x_4}) \wedge (F \overline{x_1} \vee \dots) \wedge \dots \right]$

iff $q_0 \models E \left[(X x_1 \vee X X \overline{x_2} \vee X X X X \overline{x_4}) \wedge (X \overline{x_1} \vee \dots) \wedge \dots \right]$

NP-complete path modalities

Let \mathcal{I} be a 3SAT instance. E.g. \mathcal{I} is $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \dots) \wedge \dots$



$$\begin{aligned} \mathcal{I} \text{ is satisfiable} & \text{ iff } q_0 \models E \left[(F x_1 \vee F \overline{x_2} \vee F \overline{x_4}) \wedge (F \overline{x_1} \vee \dots) \wedge \dots \right] \\ & \text{ iff } q_0 \models E \left[(X x_1 \vee X X \overline{x_2} \vee X X X X \overline{x_4}) \wedge (X \overline{x_1} \vee \dots) \wedge \dots \right] \end{aligned}$$

Model checking $E\varphi$ formulae, for $\varphi \in L^1(F)$ or $\varphi \in L(X)$ is NP-hard.

Hence model checking $B^+(F)$ or $B^*(X)$ is NP-hard and coNP-hard.

NB. Membership in NP is seen from a short witness lemma: for $\varphi \in L(F)$ or $\varphi \in L(X)$, $q \models E\varphi$ can be witnessed by a path (a loop) of length $O(|S| \times |\varphi|)$.

Model checking for CTL^+ , $FCTL$, ...

Model checking for $L^1(X, U)$ is NP-complete, thus model checking for CTL^+ is in P^{NP} .

Model checking for $L^1(\overset{\infty}{F}, X, U)$ is NP-complete, thus model checking for $ECTL^+$, $GFCTL$ and $FCTL$ is in P^{NP} .

Model checking for $L(F)$ is NP-complete, thus model checking for $B^*(F)$ is in P^{NP} .

Model checking for $L(X)$ is NP-complete, thus model checking for $B^*(X)$ is in P^{NP} .

NB. P^{NP} , a.k.a. Δ_2^p , is the class of problems that can be solved by a deterministic Turing machine that has access to an oracle in NP.

Closing the gap

Towards a complete picture

$B^*(X)$

$B^+(X)$

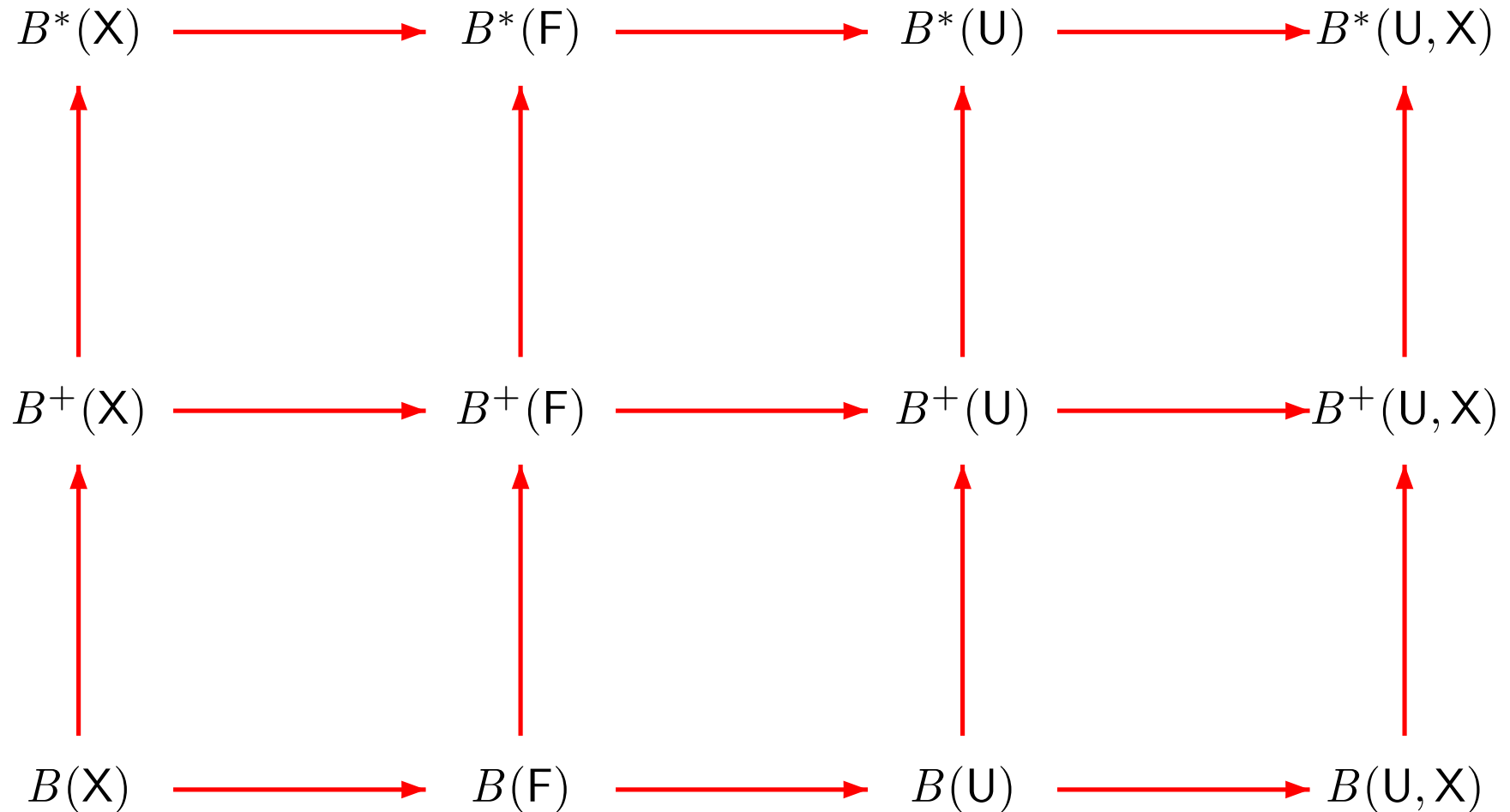
$B(X)$

$B(F)$

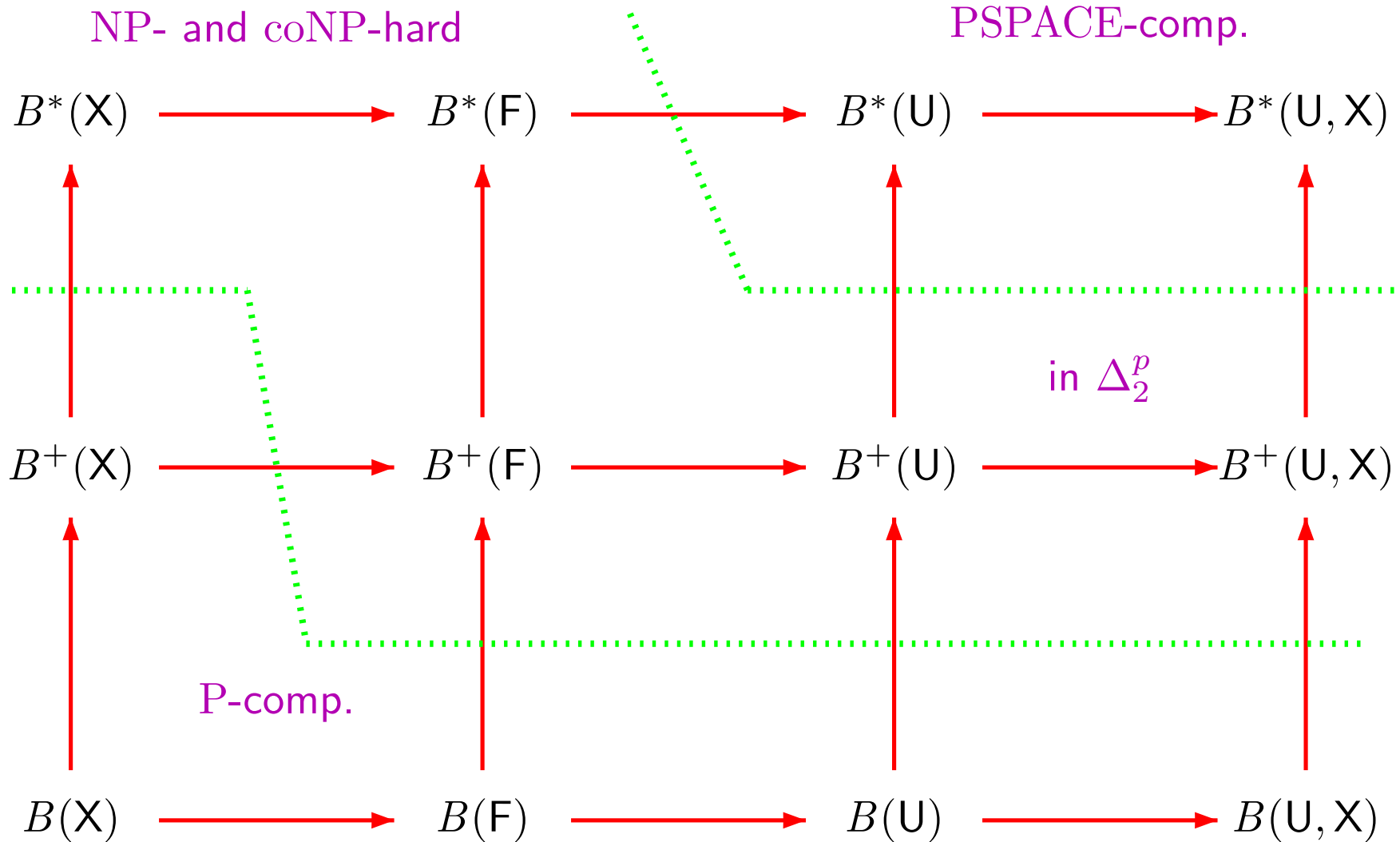
$B(U)$

$B(U, X)$

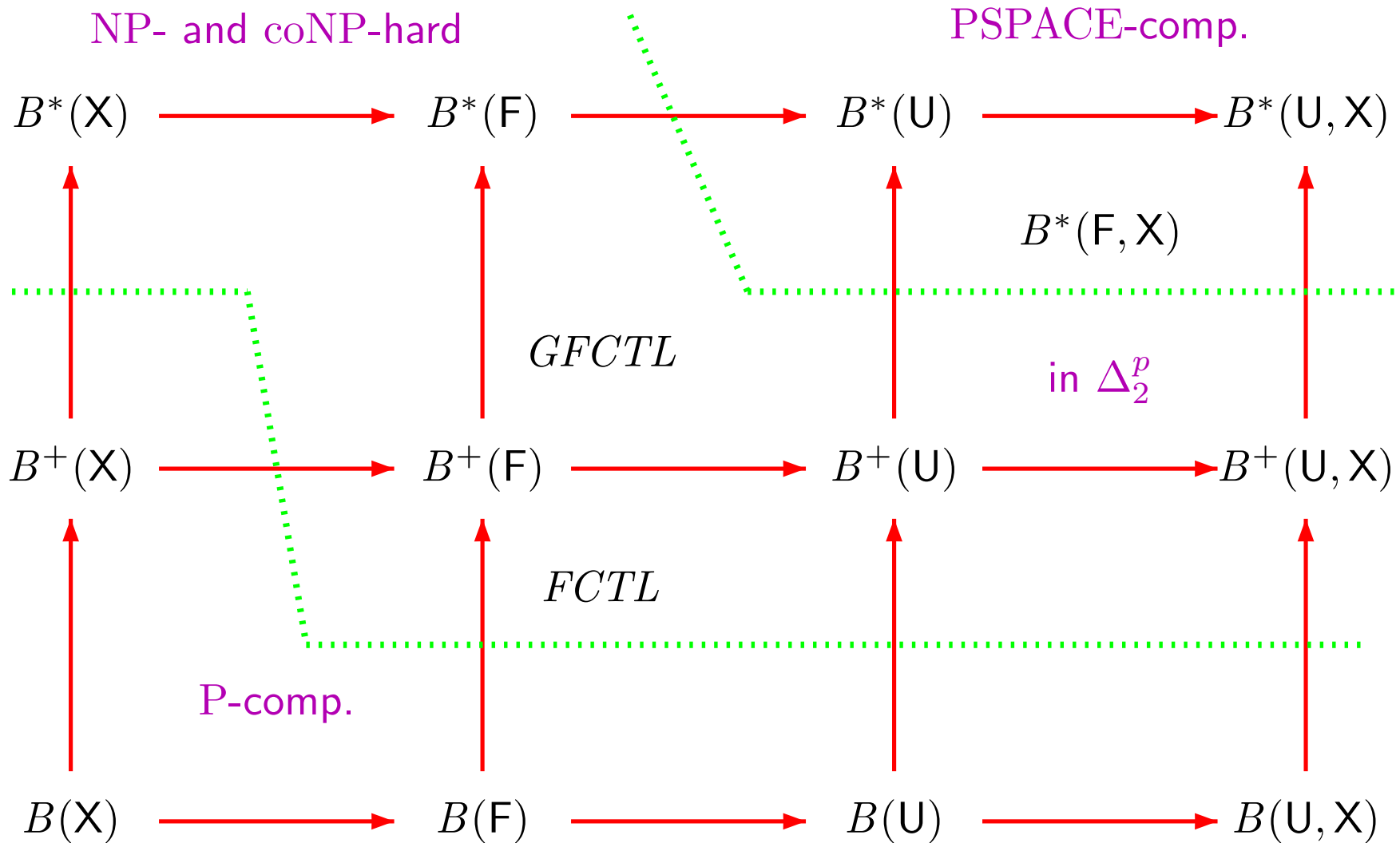
Towards a complete picture



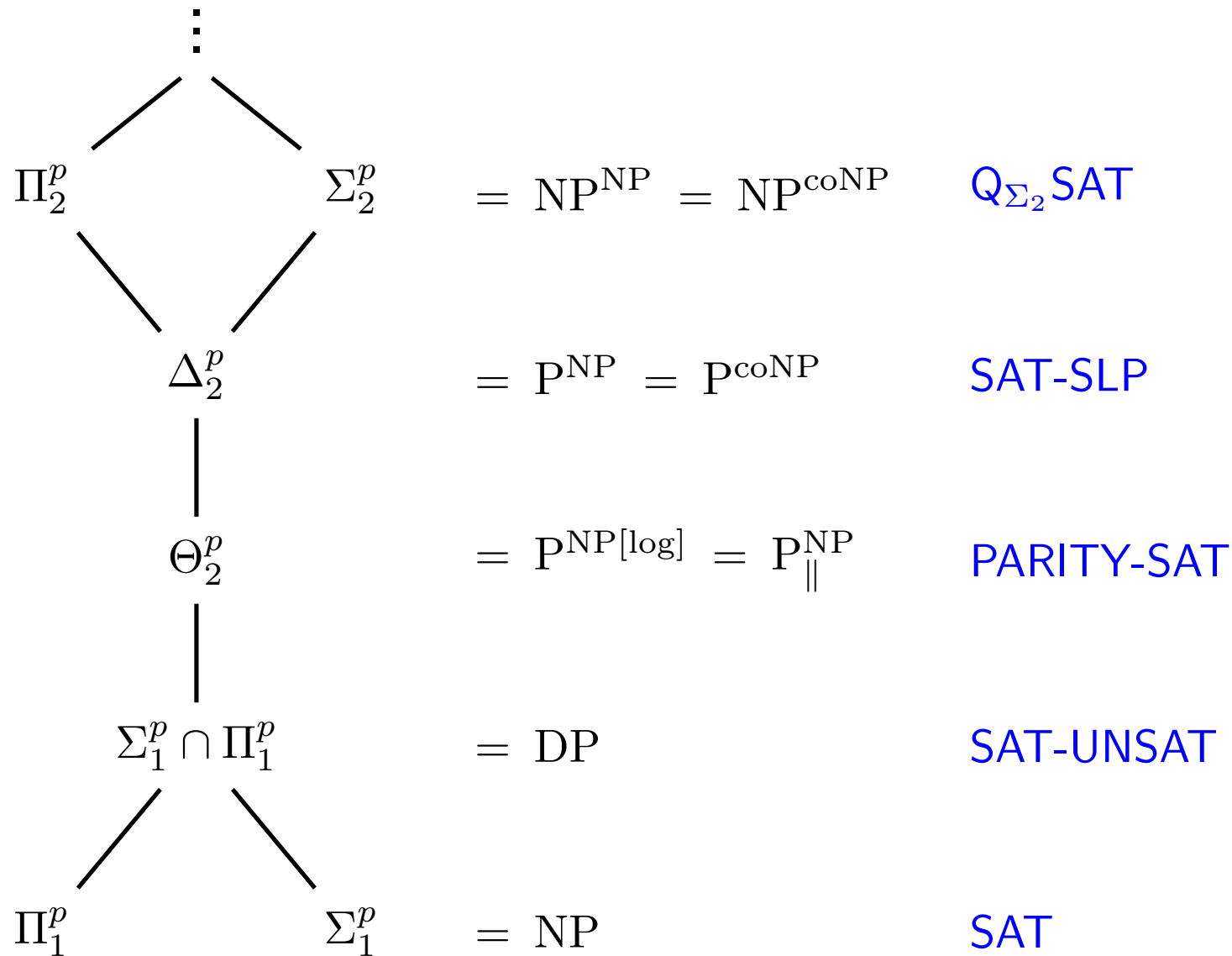
Towards a complete picture



Towards a complete picture



Δ_2^p and the polynomial-time hierarchy



Straight-line SAT programs

A **SAT-SLP** is a sequence of Boolean assignments that use adaptive SAT problems:

$$\begin{array}{ll} x_1 & :\Leftrightarrow \exists Z_1. \mathcal{B}_1(Z_1) & \text{e.g. } \exists z_1, z_2, z_3. [(z_1 \vee z_2 \vee \bar{z}_3) \wedge \dots] \\ x_2 & :\Leftrightarrow \exists Z_2. \mathcal{B}_2(x_1, Z_2) \\ x_3 & :\Leftrightarrow \exists Z_3. \mathcal{B}_3(x_1, x_2, Z_3) & \text{e.g. } x_1 \wedge x_2 \\ & \vdots \\ x_n & :\Leftrightarrow \exists Z_n. \mathcal{B}_n(x_1, x_2, \dots, x_{n-1}, Z_n) \end{array}$$

Such an SLP gives unambiguous values for the x_i 's.

The decision problem associated with the SLP is to compute x_n .

Straight-line SAT programs

A **SAT-SLP** is a sequence of Boolean assignments that use adaptive SAT problems:

$$\begin{array}{ll} x_1 & :\Leftrightarrow \exists Z_1. \mathcal{B}_1(Z_1) & \text{e.g. } \exists z_1, z_2, z_3. [(z_1 \vee z_2 \vee \bar{z}_3) \wedge \dots] \\ x_2 & :\Leftrightarrow \exists Z_2. \mathcal{B}_2(x_1, Z_2) \\ x_3 & :\Leftrightarrow \exists Z_3. \mathcal{B}_3(x_1, x_2, Z_3) & \text{e.g. } x_1 \wedge x_2 \\ & \vdots \\ x_n & :\Leftrightarrow \exists Z_n. \mathcal{B}_n(x_1, x_2, \dots, x_{n-1}, Z_n) \end{array}$$

Such an SLP gives unambiguous values for the x_i 's.

The decision problem associated with the SLP is to compute x_n .

Fact: SAT-SLP is Δ_2^P -complete.

NB: This requires that queries are adaptive in depth and width.

Δ_2^p -hardness of $B^+(F)$ model checking – 1

$$x_1 \quad :\Leftrightarrow \quad \exists Z. \mathcal{B}_1(Z)$$

$$x_2 \quad :\Leftrightarrow \quad \exists Z. \mathcal{B}_2(x_1, Z)$$

$$x_3 \quad :\Leftrightarrow \quad \exists Z. \mathcal{B}_3(x_1, x_2, Z)$$

⋮

$$x_n \quad :\Leftrightarrow \quad \exists Z. \mathcal{B}_n(x_1, \dots, x_{n-1}, Z)$$

Δ_2^p -hardness of $B^+(F)$ model checking – 1

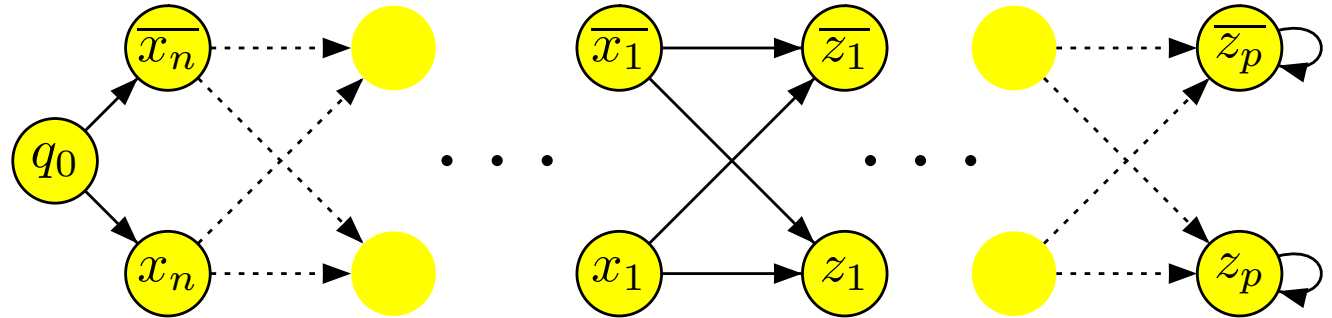
$$x_1 \Leftrightarrow \exists Z. \mathcal{B}_1(Z)$$

$$x_2 \Leftrightarrow \exists Z. \mathcal{B}_2(x_1, Z)$$

$$x_3 \Leftrightarrow \exists Z. \mathcal{B}_3(x_1, x_2, Z)$$

⋮

$$x_n \Leftrightarrow \exists Z. \mathcal{B}_n(x_1, \dots, x_{n-1}, Z)$$



Δ_2^p -hardness of $B^+(F)$ model checking – 1

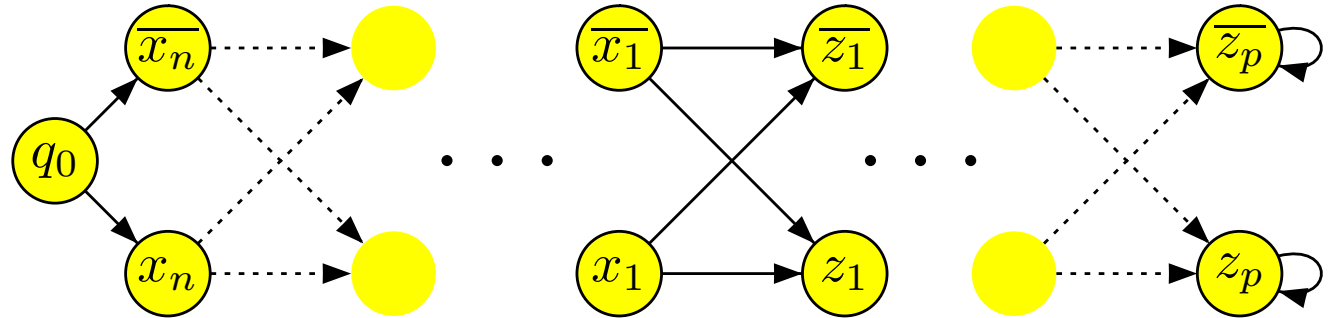
$$x_1 \Leftrightarrow \exists Z. \mathcal{B}_1(Z)$$

$$x_2 \Leftrightarrow \exists Z. \mathcal{B}_2(x_1, Z)$$

$$x_3 \Leftrightarrow \exists Z. \mathcal{B}_3(x_1, x_2, Z)$$

⋮

$$x_n \Leftrightarrow \exists Z. \mathcal{B}_n(x_1, \dots, x_{n-1}, Z)$$



$$x_1 = \mathbf{T} \text{ iff } q_0 \models \mathbf{E} \mathcal{B}_1(\mathbf{F}z_1, \dots, \mathbf{F}z_p)$$

$$“q_0 \models \varphi_1”$$

Δ_2^p -hardness of $B^+(\mathbf{F})$ model checking – 1

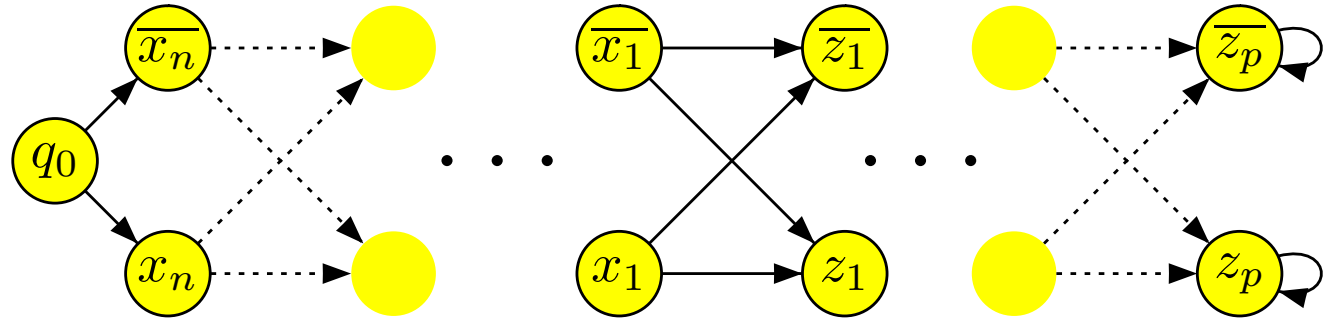
$$x_1 \Leftrightarrow \exists Z. \mathcal{B}_1(Z)$$

$$x_2 \Leftrightarrow \exists Z. \mathcal{B}_2(x_1, Z)$$

$$x_3 \Leftrightarrow \exists Z. \mathcal{B}_3(x_1, x_2, Z)$$

⋮

$$x_n \Leftrightarrow \exists Z. \mathcal{B}_n(x_1, \dots, x_{n-1}, Z)$$



$$x_1 = \mathbf{T} \text{ iff } q_0 \models \mathbf{E} \mathcal{B}_1(\mathbf{F}z_1, \dots, \mathbf{F}z_p)$$

$$"q_0 \models \varphi_1"$$

$$x_2 = \mathbf{T} \text{ iff } q_0 \models \mathbf{E} \left[\begin{array}{l} \mathcal{B}_2(\mathbf{F}x_1, \mathbf{F}z_1, \dots, \mathbf{F}z_p) \\ \wedge (\mathbf{F}x_1 \Leftrightarrow \varphi_1) \end{array} \right]$$

$$"q_0 \models \varphi_2"$$

Δ_2^p -hardness of $B^+(\mathbf{F})$ model checking – 1

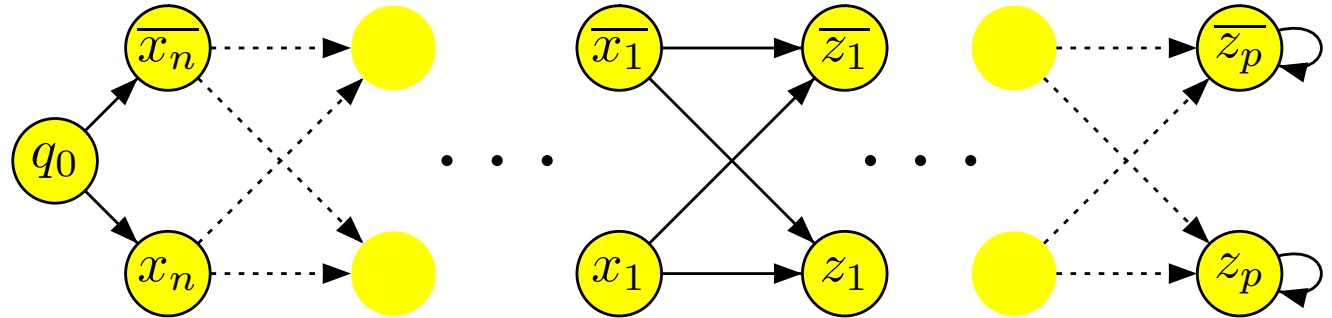
$$x_1 \Leftrightarrow \exists Z. \mathcal{B}_1(Z)$$

$$x_2 \Leftrightarrow \exists Z. \mathcal{B}_2(x_1, Z)$$

$$x_3 \Leftrightarrow \exists Z. \mathcal{B}_3(x_1, x_2, Z)$$

⋮

$$x_n \Leftrightarrow \exists Z. \mathcal{B}_n(x_1, \dots, x_{n-1}, Z)$$



$$x_1 = \mathbf{T} \text{ iff } q_0 \models \mathbf{E} \mathcal{B}_1(\mathbf{F}z_1, \dots, \mathbf{F}z_p)$$

“ $q_0 \models \varphi_1$ ”

$$x_2 = \mathbf{T} \text{ iff } q_0 \models \mathbf{E} \left[\begin{array}{l} \mathcal{B}_2(\mathbf{F}x_1, \mathbf{F}z_1, \dots, \mathbf{F}z_p) \\ \wedge (\mathbf{F}x_1 \Leftrightarrow \varphi_1) \end{array} \right]$$

“ $q_0 \models \varphi_2$ ”

⋮

$$x_n = \mathbf{T} \text{ iff } q_0 \models \mathbf{E} \left[\begin{array}{l} \mathcal{B}_n(\mathbf{F}x_1, \dots, \mathbf{F}x_{n-1}, \mathbf{F}z_1, \dots, \mathbf{F}z_p) \\ \wedge (\mathbf{F}x_1 \Leftrightarrow \varphi_1) \wedge \dots \wedge (\mathbf{F}x_{n-1} \Leftrightarrow \varphi_{n-1}) \end{array} \right]$$

“ $q_0 \models \varphi_n$ ”

Δ_2^p -hardness of $B^+(F)$ model checking – 1

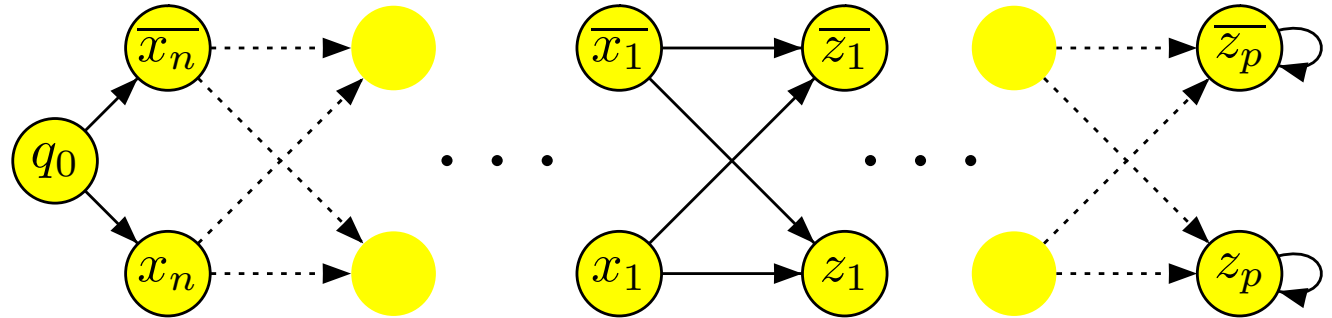
$$x_1 :\Leftrightarrow \exists Z. \mathcal{B}_1(Z)$$

$$x_2 :\Leftrightarrow \exists Z. \mathcal{B}_2(x_1, Z)$$

$$x_3 :\Leftrightarrow \exists Z. \mathcal{B}_3(x_1, x_2, Z)$$

⋮

$$x_n :\Leftrightarrow \exists Z. \mathcal{B}_n(x_1, \dots, x_{n-1}, Z)$$



$$x_1 = \text{T} \text{ iff } q_0 \models \text{E } \mathcal{B}_1(\text{F}z_1, \dots, \text{F}z_p)$$

“ $q_0 \models \varphi_1$ ”

$$x_2 = \text{T} \text{ iff } q_0 \models \text{E} \left[\begin{array}{l} \mathcal{B}_2(\text{F}x_1, \text{F}z_1, \dots, \text{F}z_p) \\ \wedge (\text{F}x_1 \Leftrightarrow \varphi_1) \end{array} \right]$$

“ $q_0 \models \varphi_2$ ”

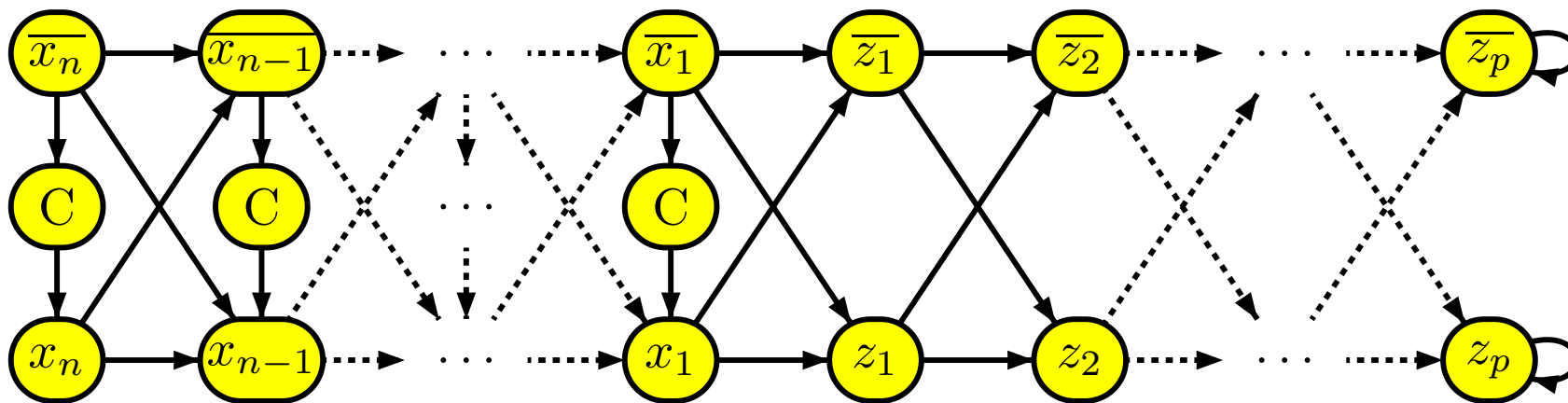
⋮

$$x_n = \text{T} \text{ iff } q_0 \models \text{E} \left[\begin{array}{l} \mathcal{B}_n(\text{F}x_1, \dots, \text{F}x_{n-1}, \text{F}z_1, \dots, \text{F}z_p) \\ \wedge (\text{F}x_1 \Leftrightarrow \varphi_1) \wedge \dots \wedge (\text{F}x_{n-1} \Leftrightarrow \varphi_{n-1}) \end{array} \right]$$

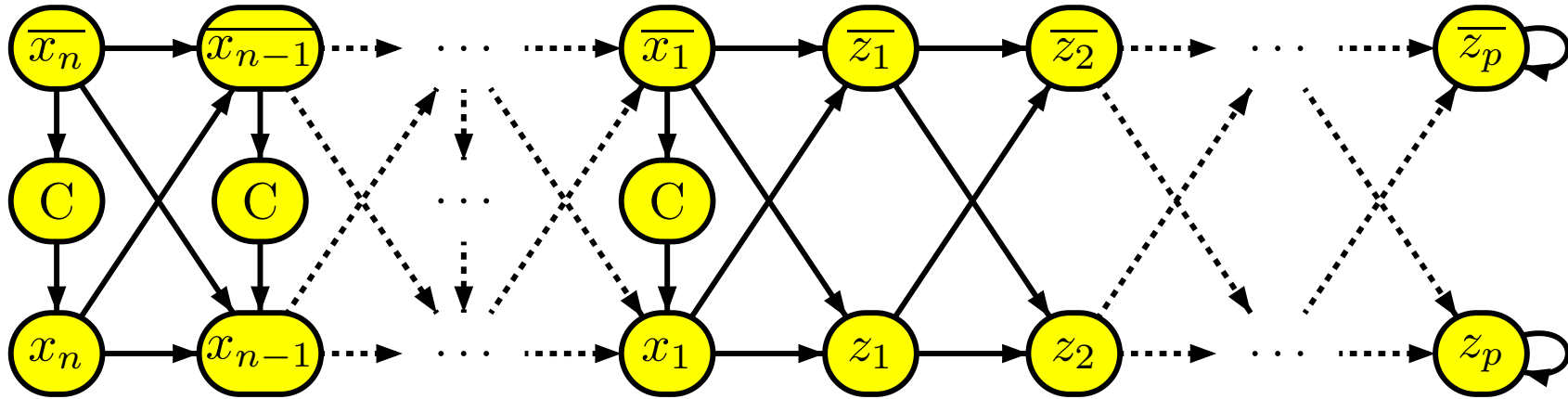
“ $q_0 \models \varphi_n$ ”

Pb. φ_n has exponential size!

Δ_2^p -hardness of $B^+(F)$ model checking – 2

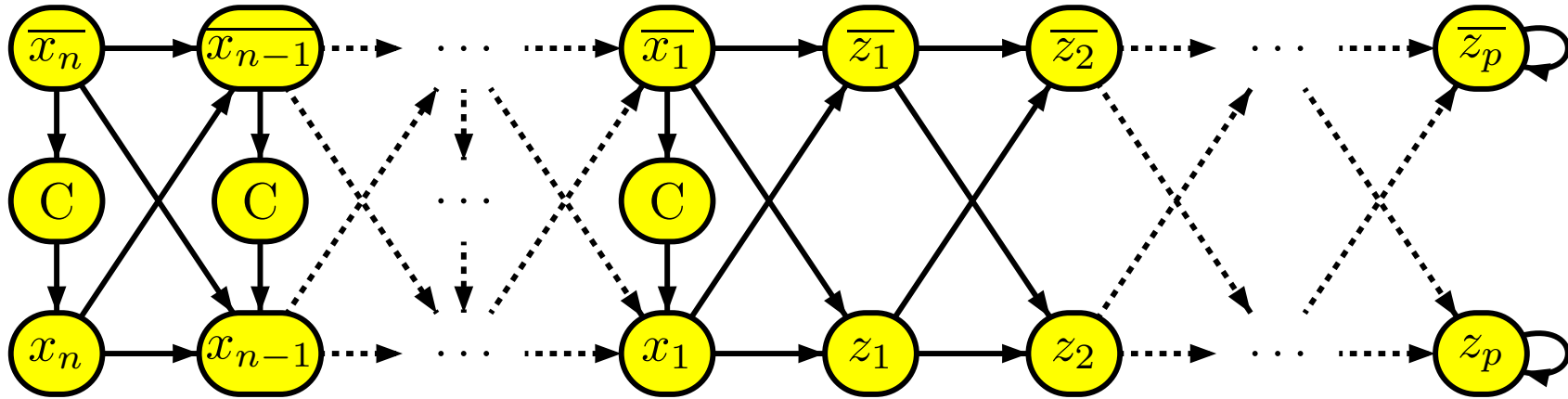


Δ_2^p -hardness of $B^+(F)$ model checking – 2



$$\varphi = E \left[\begin{array}{l} G \neg C \\ \wedge \bigwedge_{i=1}^n [F x_i \Rightarrow \mathcal{B}_i(F x_1, \dots, F x_{i-1}, F z_1, \dots, F z_p)] \\ \wedge G [(\overline{x_1} \vee \dots \vee \overline{x_n}) \Rightarrow \text{EX}(C \wedge \text{EX} \neg \varphi)] \end{array} \right]$$

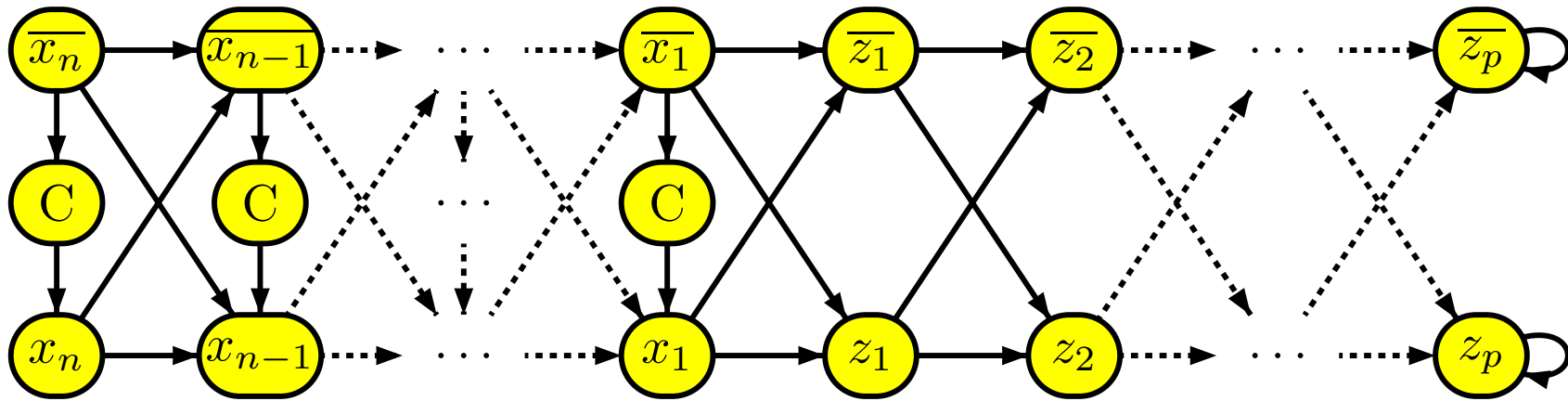
Δ_2^p -hardness of $B^+(F)$ model checking – 2



$$\varphi = E \left[\begin{array}{l} G \neg C \\ \wedge \bigwedge_{i=1}^n [F x_i \Rightarrow \mathcal{B}_i(F x_1, \dots, F x_{i-1}, F z_1, \dots, F z_p)] \\ \wedge G [(\overline{x}_1 \vee \dots \vee \overline{x}_n) \Rightarrow EX(C \wedge EX \neg \varphi)] \end{array} \right]$$

Unfold the definition with $\varphi_0 = \top$ and $\varphi_{i+1} = E[\dots \wedge G[\dots EX \neg \varphi_i]]$.

Δ_2^p -hardness of $B^+(F)$ model checking – 2



$$\varphi = E \left[\begin{array}{l} G \neg C \\ \wedge \bigwedge_{i=1}^n [F x_i \Rightarrow \mathcal{B}_i(F x_1, \dots, F x_{i-1}, F z_1, \dots, F z_p)] \\ \wedge G [(\bar{x}_1 \vee \dots \vee \bar{x}_n) \Rightarrow EX(C \wedge EX \neg \varphi)] \end{array} \right]$$

Unfold the definition with $\varphi_0 = \mathbf{T}$ and $\varphi_{i+1} = E[\dots \wedge G[\dots EX \neg \varphi_i]]$.

Fact. $x_n = \mathbf{T}$ in the SAT-SLP iff $x_n \models \varphi_{2n-1}$ in the Kripke structure.

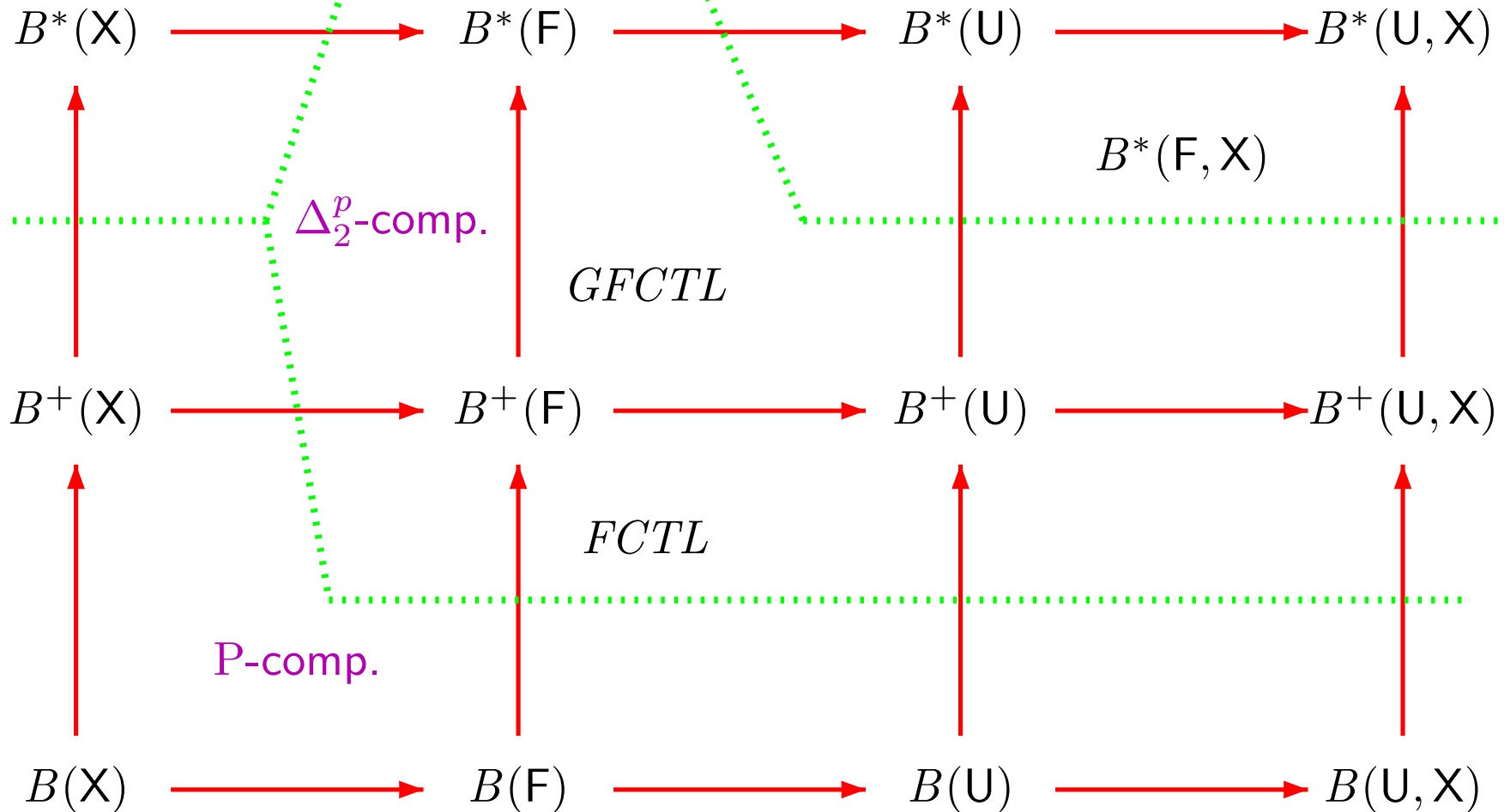
Hence Δ_2^p -hardness of $B^+(F)$ model checking.

(Similar tricks show Δ_2^p -hardness of $B^+(\overset{\infty}{F})$ model checking.)

An almost complete picture

NP- and coNP-hard

PSPACE-comp.



Last step: $B^*(X)$ [S. 2003]

What about $B^*(X)$?

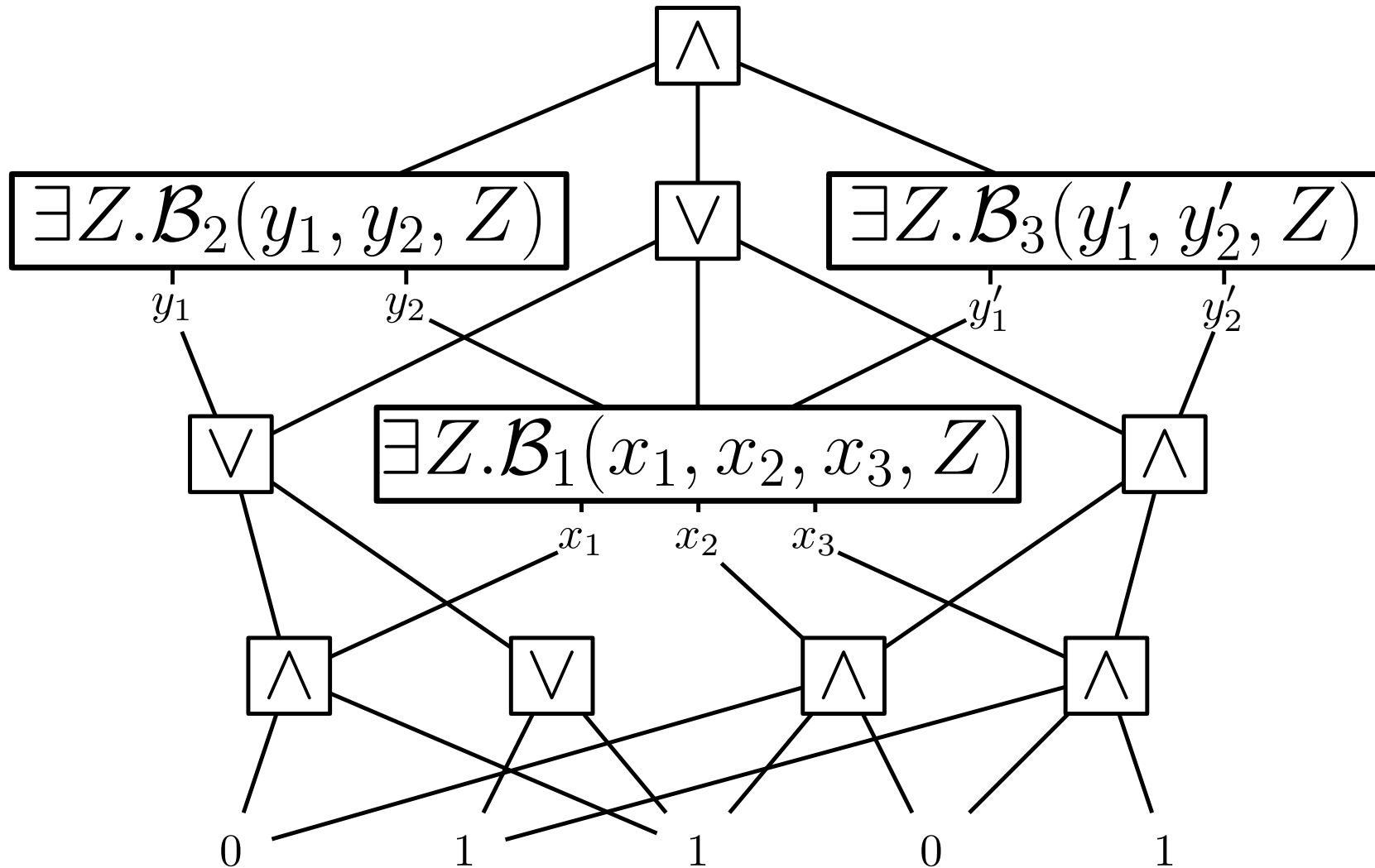
$B^*(X)$ has the same expressive power as $B(X)$ (a.k.a. ML) but is exponentially more succinct.

Previous techniques unable to show Δ_2^p -hardness of model checking $B^*(X)$.

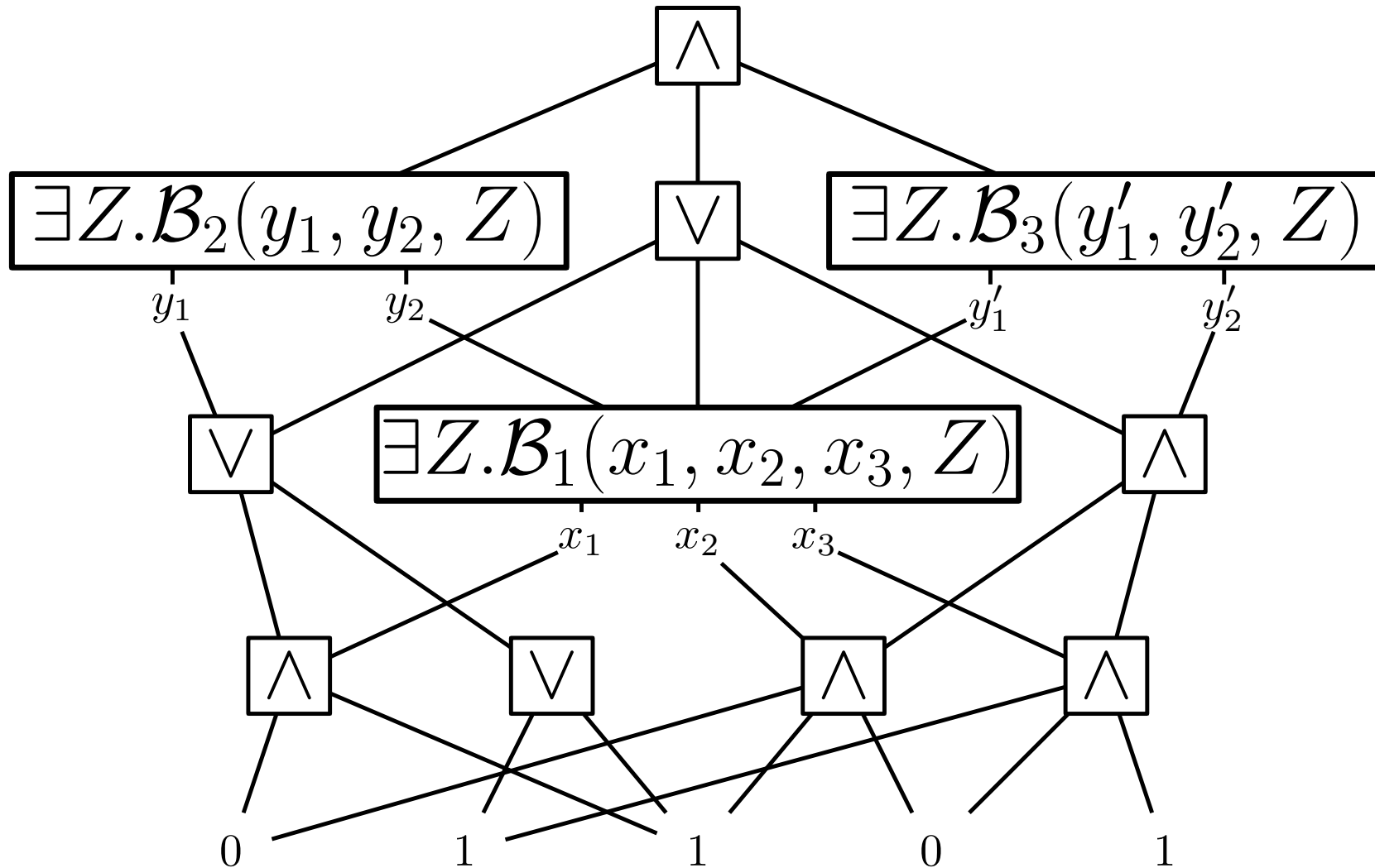
Reason is that $B^*(X)$ model checking translates into SAT-SLP's with special patterns of dependencies between queries.

This is better studied via the framework of [circuits with SAT queries](#).

Circuits with SAT queries

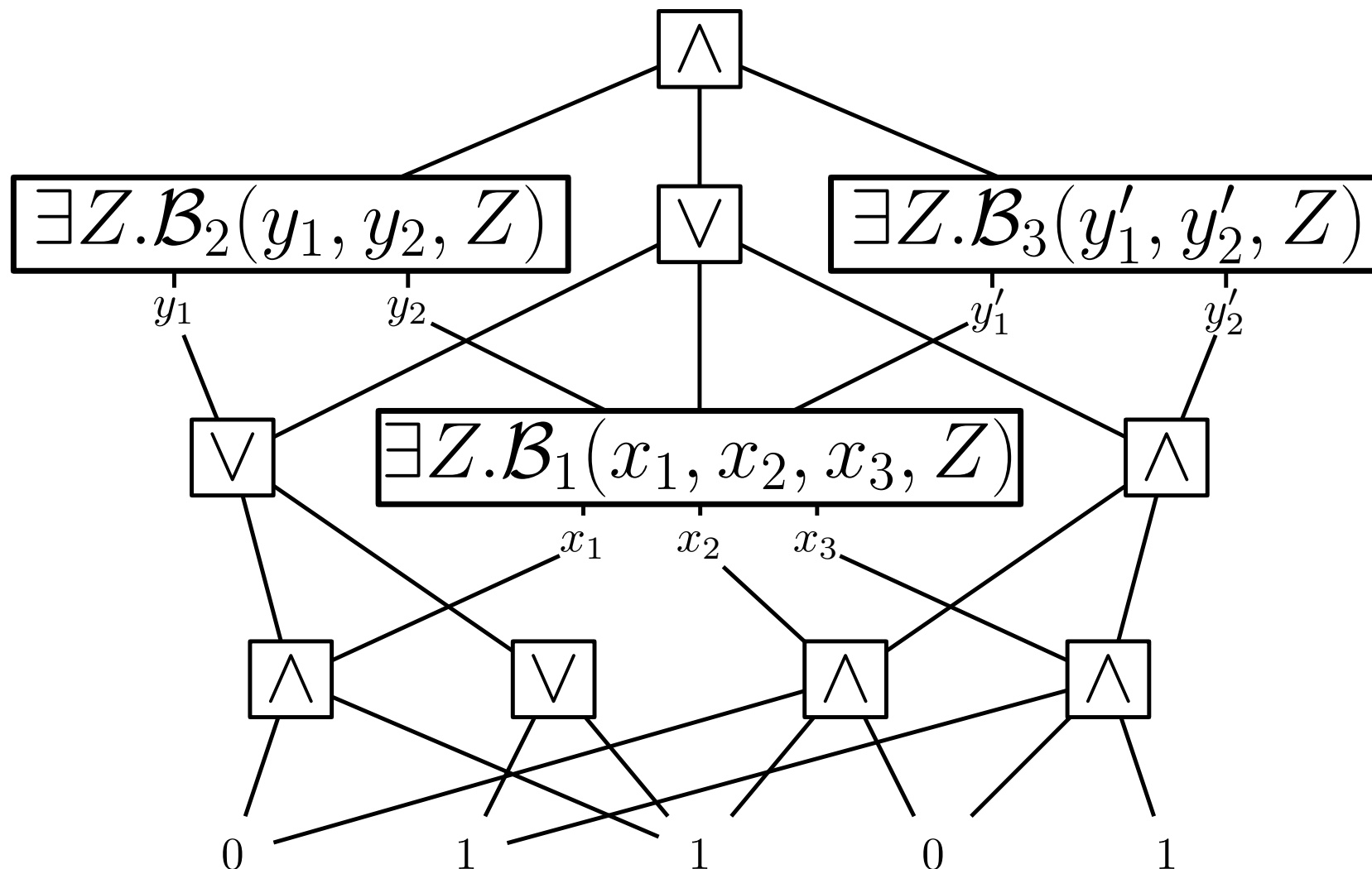


Circuits with SAT queries



Fact: DAG-SAT is Δ_2^p -complete

Circuits with SAT queries

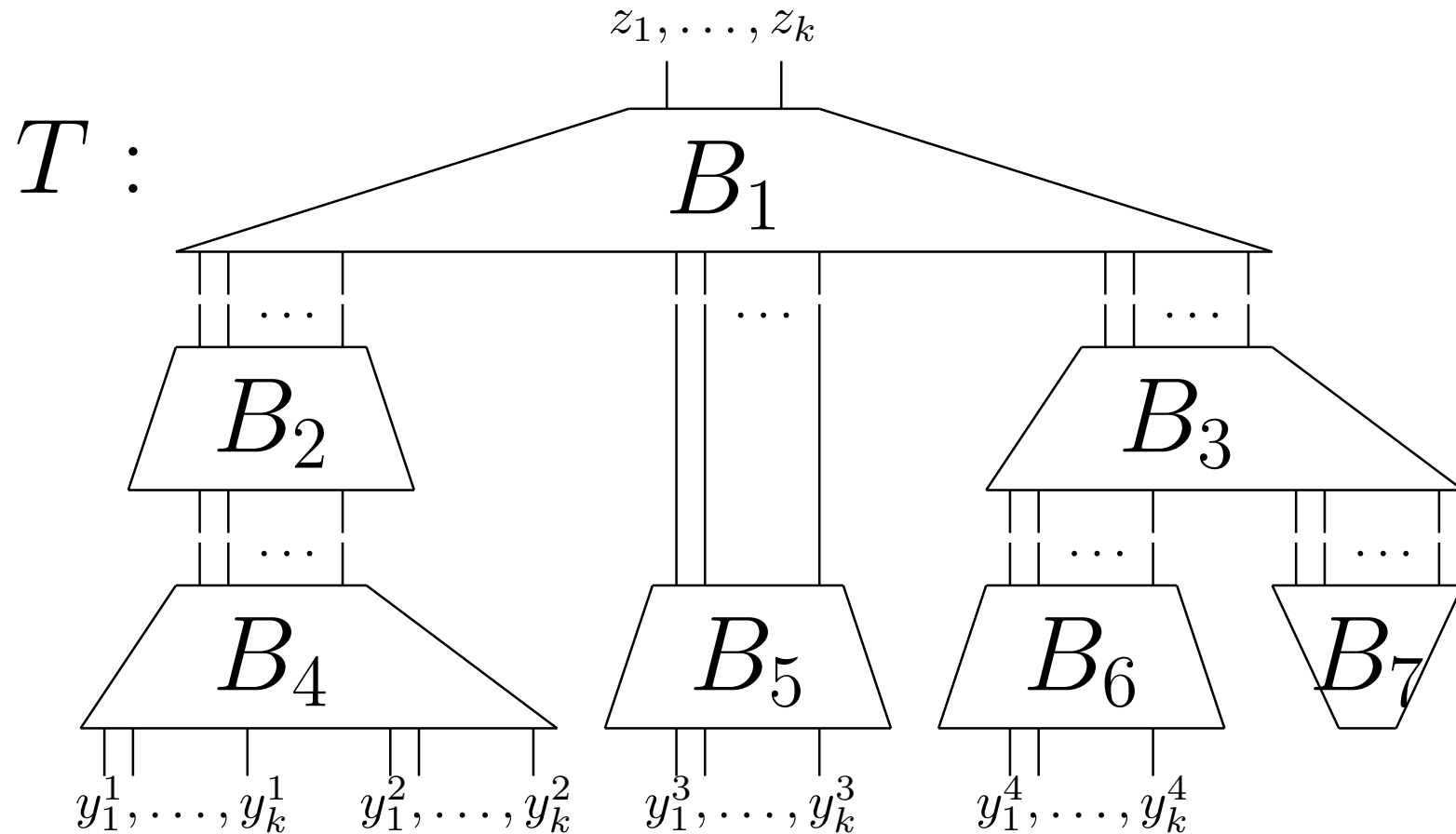


Fact: DAG-SAT is Δ_2^p -complete

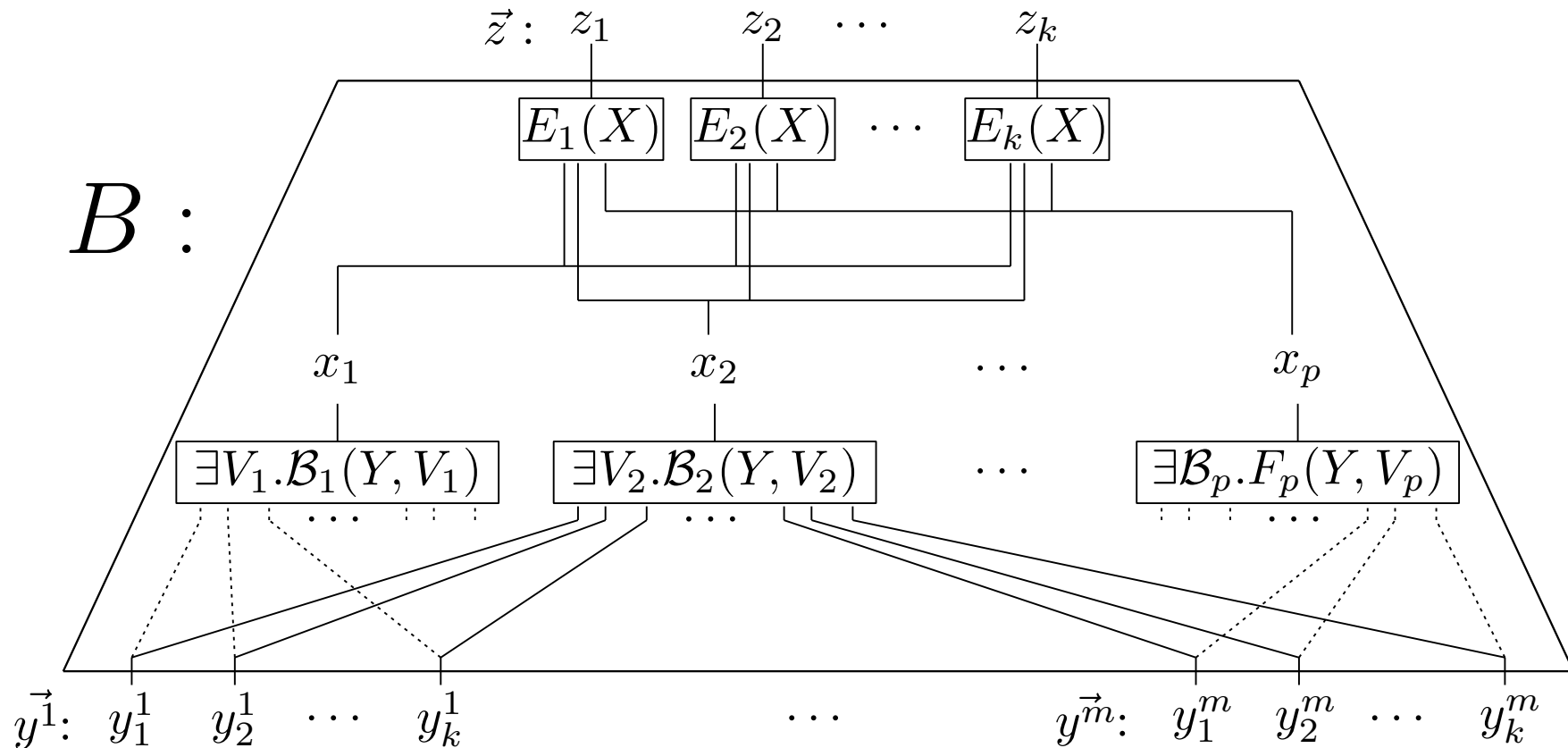
NB: TREE-SAT is Θ_2^p -complete [Gottlob 1995]

TV-SAT : trees of blocks

Model checking for $B(L)$ translates into a tree of “blocks” containing parallel queries to a model checker for L .



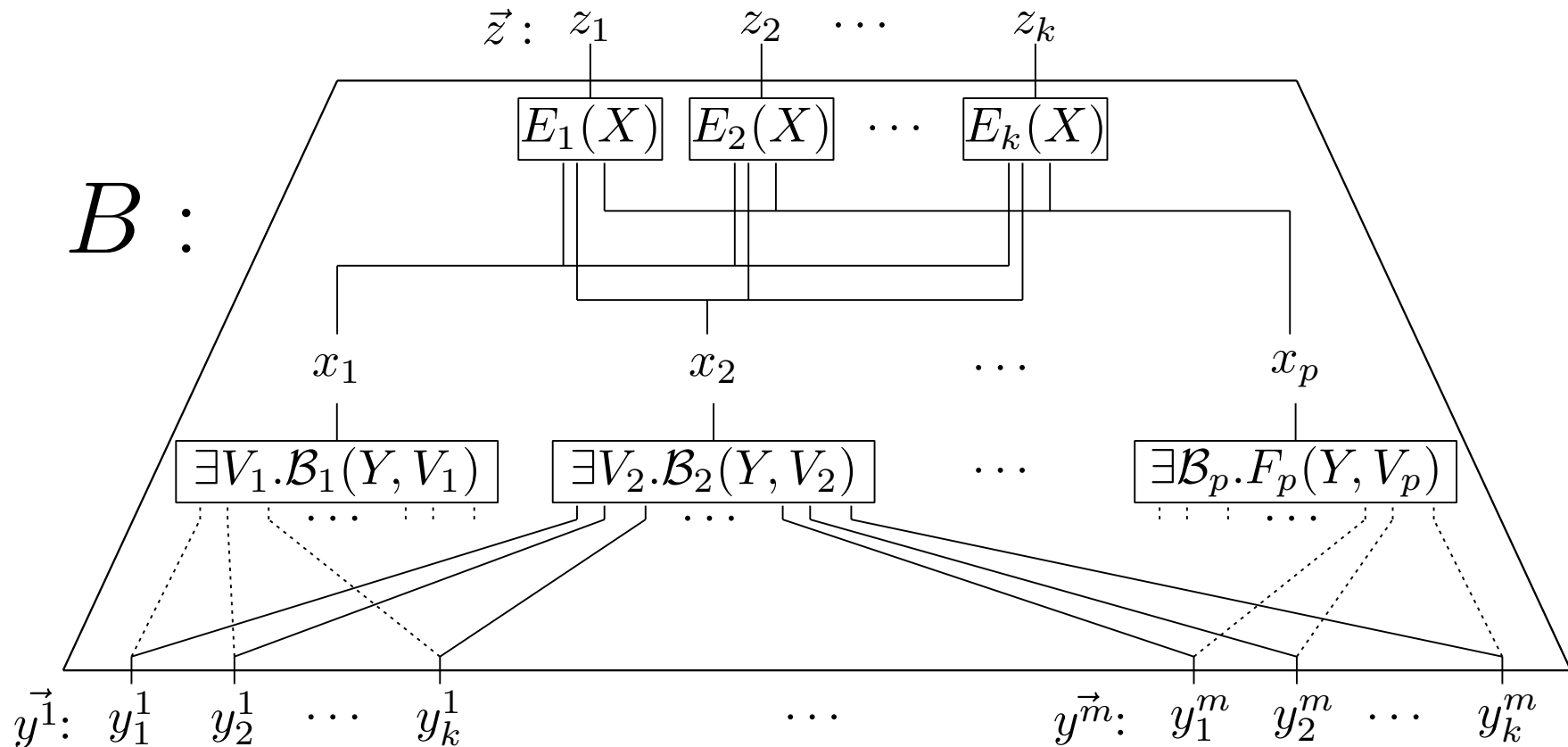
Blocks with parallel SAT queries



A block like B

- inputs m vectors of k bits each,
- combines parallel SAT queries and some combinatory logic,
- outputs one vector of k bits.

Blocks with parallel SAT queries



A block like B

- inputs m vectors of k bits each,
- combines parallel SAT queries and some combinatory logic,
- outputs one vector of k bits.

Fact. TV-SAT (i.e., evaluating trees of blocks) is Δ_2^p -complete

$B^*(X)$ translates into type 1xM queries

$$q_i \models E(X((Xa) \Rightarrow (XXXb)) \vee XX\neg c)$$

$$q_i \models E(X^2a \Rightarrow X^4b \vee \neg X^2c)$$

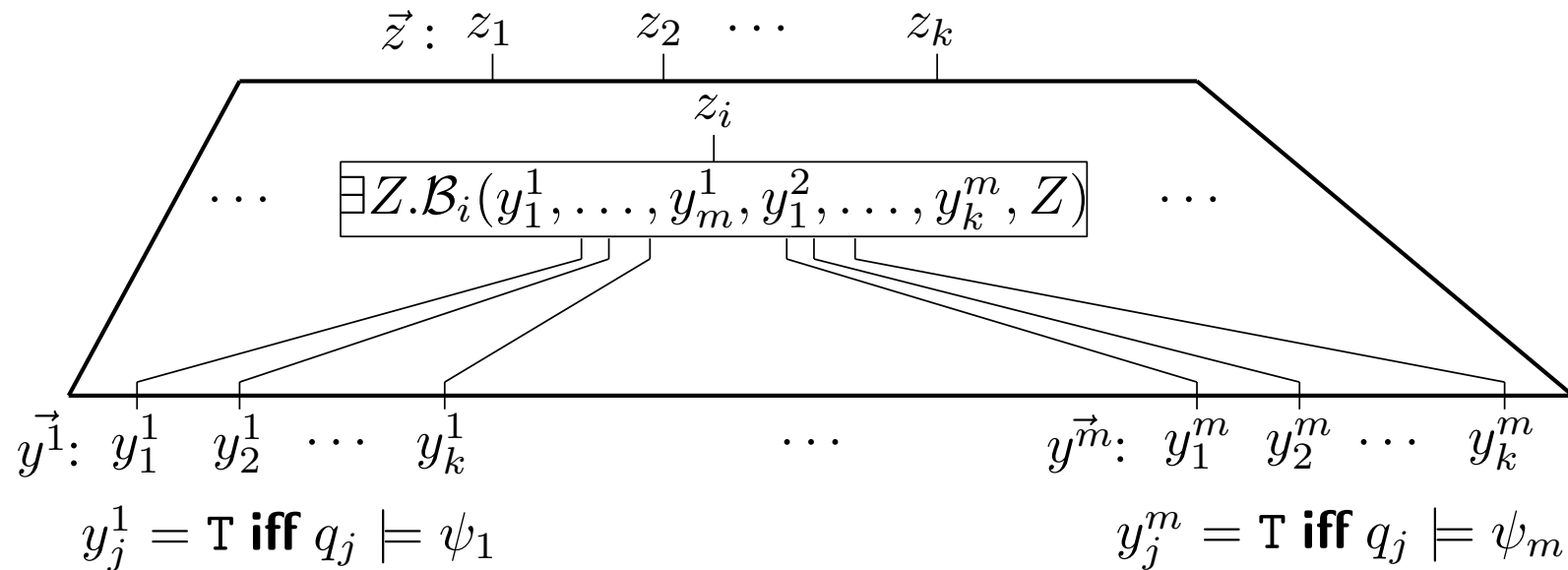
$$z_i = \mathbf{T} \text{ iff } q_i \models E\mathcal{B}(X^{n_1}\psi_1, \dots, X^{n_m}\psi_m)$$

$B^*(X)$ translates into type 1xM queries

$$q_i \models E(X((Xa) \Rightarrow (XXXb)) \vee \neg X\neg c)$$

$$q_i \models E(X^2a \Rightarrow X^4b \vee \neg X^2c)$$

$$z_i = \mathbf{T} \text{ iff } q_i \models EB(X^{n_1}\psi_1, \dots, X^{n_m}\psi_m)$$

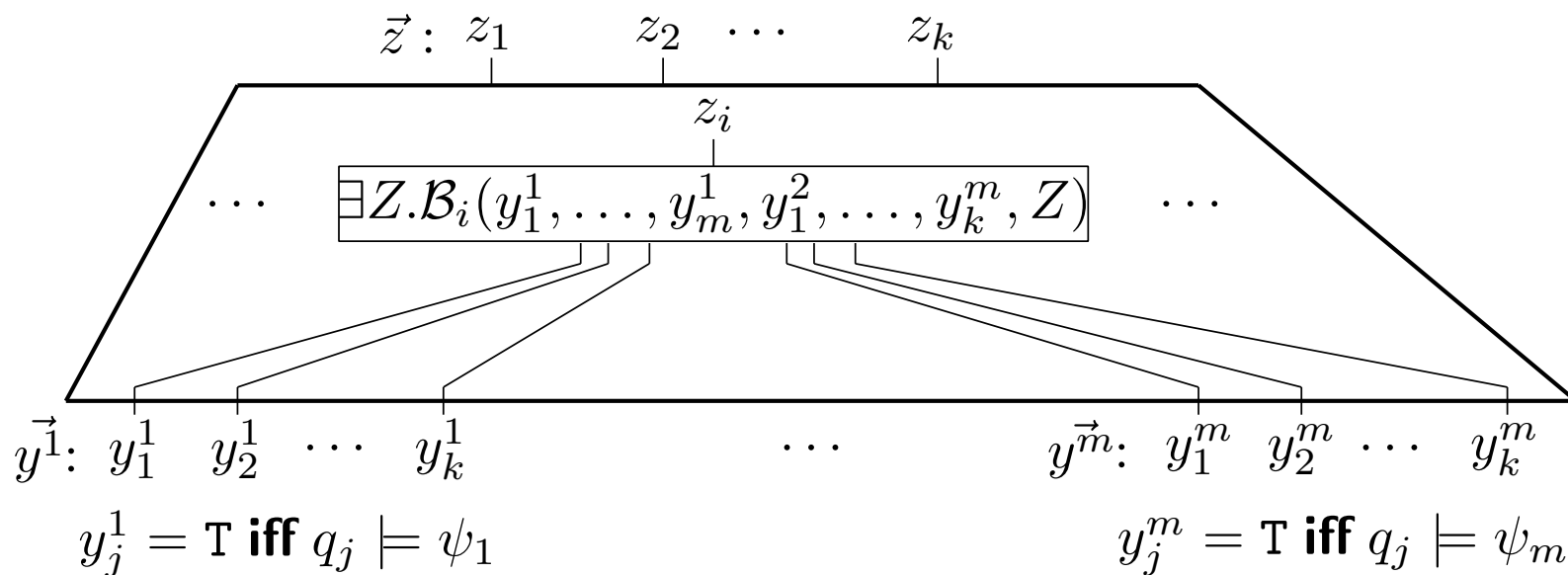


$B^*(X)$ translates into type 1xM queries

$$q_i \models E(X((Xa) \Rightarrow (XXXb)) \vee XX\neg c)$$

$$q_i \models E(X^2a \Rightarrow X^4b \vee \neg X^2c)$$

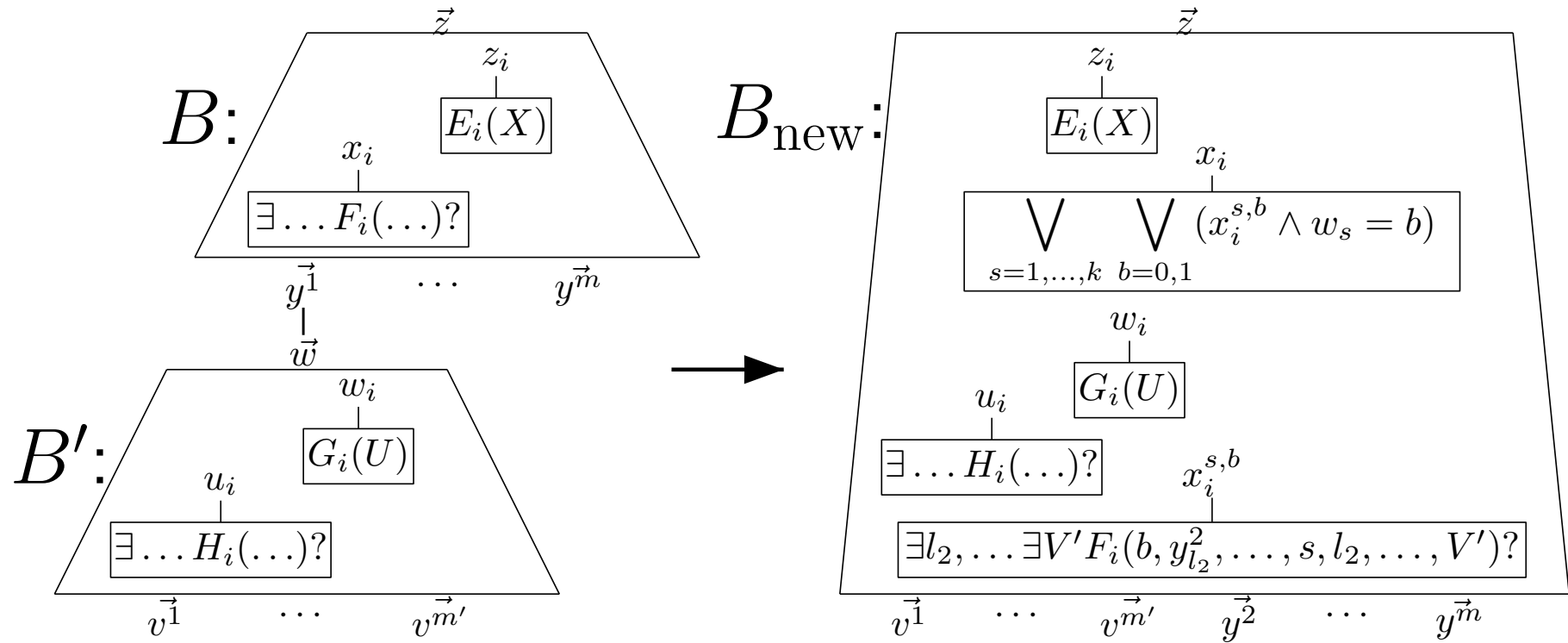
$$z_i = \mathbf{T} \text{ iff } q_i \models E\mathcal{B}(X^{n_1}\psi_1, \dots, X^{n_m}\psi_m)$$



$B^*(X)$ model-checking only needs trees of blocks with special “type 1xM” queries.

$$\exists l_1, \dots, l_m, Z'. \mathcal{B}'_i(y_{l_1}^1, \dots, y_{l_m}^m, l_1, \dots, l_m, Z')$$

Flattening trees with type 1xM queries – 1



$$|B_{\text{new}}| = O(|B'| + k|B|).$$

Flattening trees with type 1xM queries – 2

TV-SAT trees with type 1xM queries can be transformed into “balanced” trees where every node at some height h has at least two children at height $h - 1$. (This transformation is logspace.)

Hence these trees can be transformed into equivalent trees of logarithmic height, i.e. into trees with $O(\log n)$ nesting of queries.

Flattening trees with type 1xM queries – 2

TV-SAT trees with type 1xM queries can be transformed into “balanced” trees where every node at some height h has at least two children at height $h - 1$. (This transformation is logspace.)

Hence these trees can be transformed into equivalent trees of logarithmic height, i.e. into trees with $O(\log n)$ nesting of queries.

Using $\text{P}^{\text{NP}[\log^{k+1}]} = \text{P}_{\parallel[\log^k]}^{\text{NP}}$ [Castro & Seara 1996] we obtain:

Flattening trees with type 1xM queries – 2

TV-SAT trees with type 1xM queries can be transformed into “balanced” trees where every node at some height h has at least two children at height $h - 1$. (This transformation is logspace.)

Hence these trees can be transformed into equivalent trees of logarithmic height, i.e. into trees with $O(\log n)$ nesting of queries.

Using $\text{P}^{\text{NP}[\log^{k+1}]} = \text{P}_{\parallel[\log^k]}^{\text{NP}}$ [Castro & Seara 1996] we obtain:

Theo. Model checking $B^*(X)$ is in $\text{P}^{\text{NP}[\log^2]}$.

Flattening trees with type 1xM queries – 2

TV-SAT trees with type 1xM queries can be transformed into “balanced” trees where every node at some height h has at least two children at height $h - 1$. (This transformation is logspace.)

Hence these trees can be transformed into equivalent trees of logarithmic height, i.e. into trees with $O(\log n)$ nesting of queries.

Using $\text{P}^{\text{NP}[\log^{k+1}]} = \text{P}_{\parallel[\log^k]}^{\text{NP}}$ [Castro & Seara 1996] we obtain:

Theo. Model checking $B^*(X)$ is in $\text{P}^{\text{NP}[\log^2]}$.

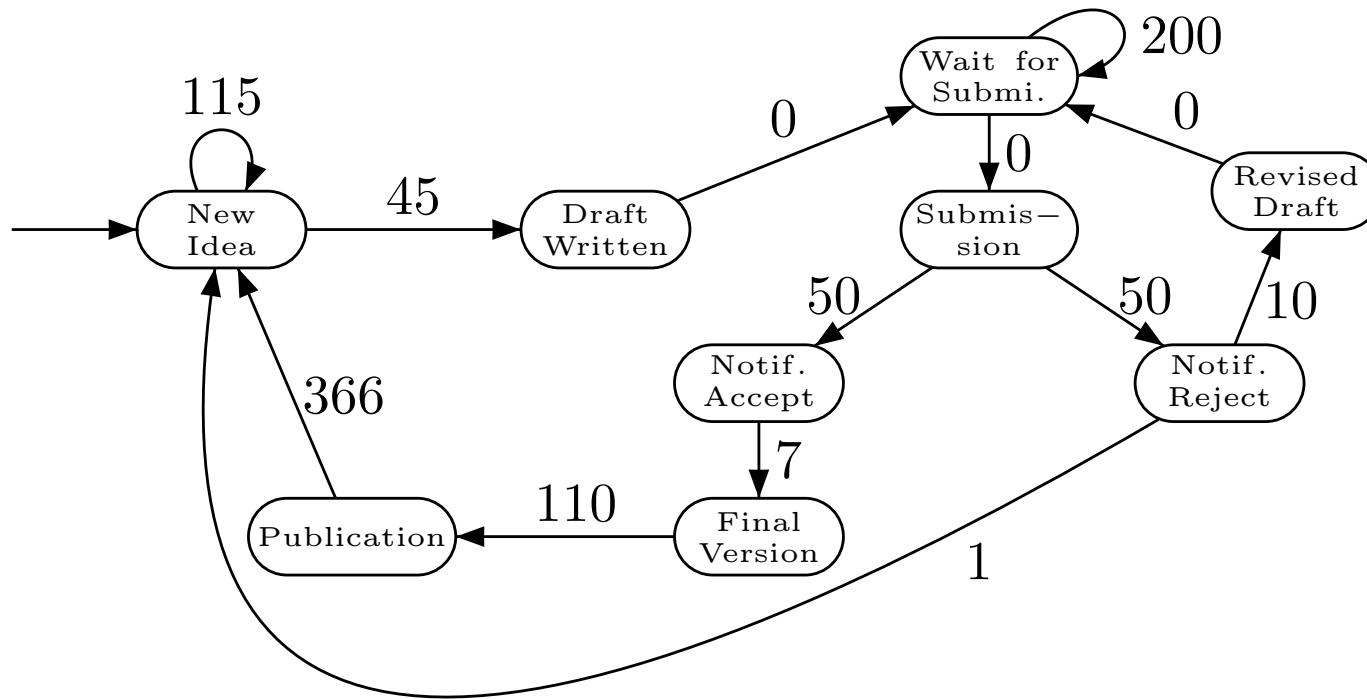
Theo. It is in fact complete for $\text{P}^{\text{NP}[\log^2]}$.

Conclusions & Perspectives

- First apparition of $P^{NP[\log^2]}$ in the literature
- Model checking problems for branching-time logics have a specific structure
(That's why the techniques apply more generally)
- Counting how many times the SAT oracle has to be invoked uses new techniques
(Perhaps tree automata are not fine-grained enough?)

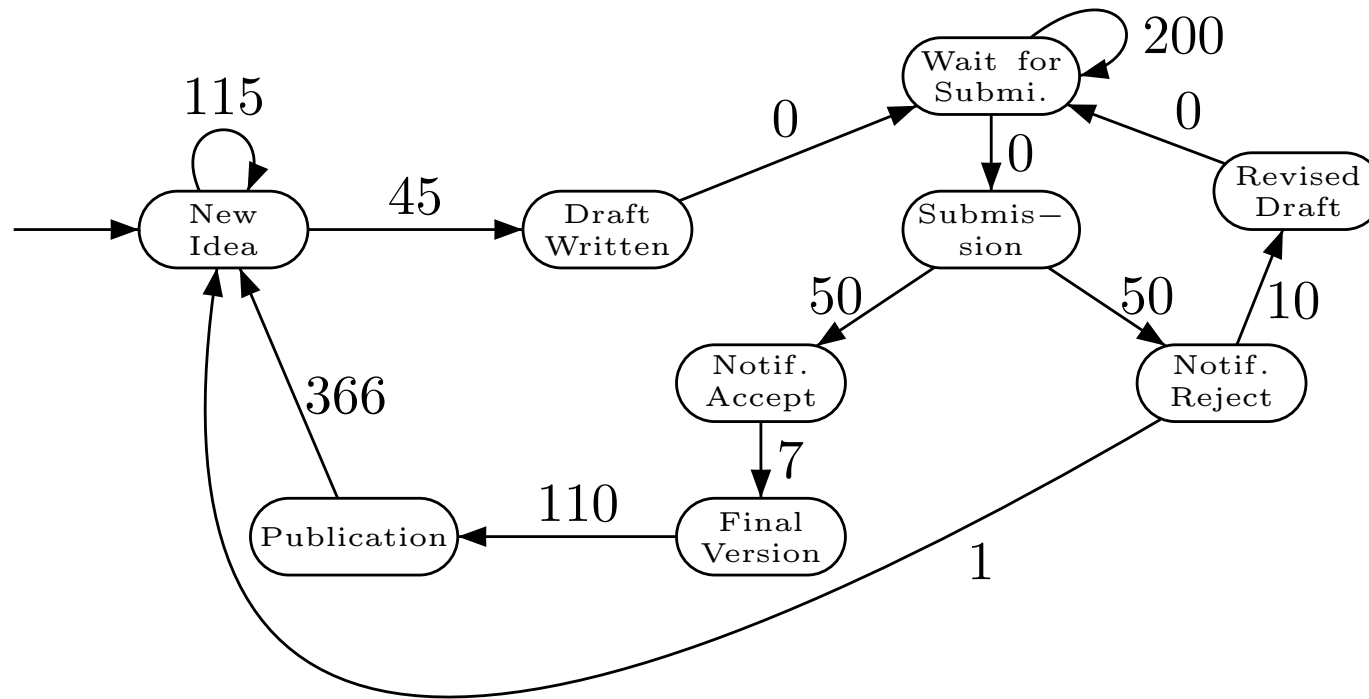
Applies to other branching-time logics

TCTL, a.k.a. *Timed-CTL*, allows $E a U_{=80}(AF_{\leq 35} b)$.



Applies to other branching-time logics

TCTL, a.k.a. *Timed-CTL*, allows $E a U_{=80}(AF_{\leq 35} b)$.



Theo: [Laroussinie, Markey, S. FoSSaCS'2002, S. 2003]

- Model checking *TCTL* on graphs with durations is Δ_2^p -complete [LMS 02]
- Model checking the $B(F)$ fragment is Θ_2^p -complete [Sch 03]