Model Checking Branching-Time Logics

Ph. Schnoebelen

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

Lab. Specification & Verification (LSV)
ENS de Cachan & CNRS
Cachan, France
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 $\Delta^p_2$
- Completing the picture
- Conclusions and perspectives
Classics from branching-time model checking
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:

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

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 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 $AG[\langle X(EF\text{restart}) \rangle UX\text{stop}]$.

**Example:** $CTL^*$ is $B(LTL)$. 
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 $AG[(X(EF_{\text{restart}})UX\text{ stop})]$.

**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]

$$
\phi = AX A(\Box \neg a \land \Diamond \neg AF(Xb \land \neg a)))
$$
Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\Box \neg a \land \Diamond \neg AF(Xb \land \neg a)))$$

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

![Diagram of a model checking for $B(L)$](image)
Model checking $B(L)$ [Emerson & Lei, 1987]

$\varphi = AX A(F\neg a \land G \neg AF(Xb \land \neg a))$

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

$s_2 \models A \varphi_1$

$s_1, s_3, s_4 \not\models A \varphi_1$
\( \varphi = \Box X A(\neg F a \land \neg \Diamond AF(Xb \land \neg a))) \)

\( \varphi_1 = F(Xb \land \neg a) \)

\( s_2 \models A \varphi_1 \)

\( s_1, s_3, s_4 \not\models A \varphi_1 \)
Model checking $B(L)$ [Emerson & Lei, 1987]

$$\varphi = AX A(\Box \neg a \land \Box \neg AF(Xb \land \neg a))$$

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

$$s_2 \models A \varphi_1$$

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

$$\varphi_2 = \Box \neg a \land \Box \neg P_1$$
\[ \varphi = AX A(\Diamond \neg a \land \Box \neg AF(Xb \land \neg a)) \]

\[ \varphi_1 = F(Xb \land \neg a) \]
\[ s_2 \models A \varphi_1 \]
\[ s_1, s_3, s_4 \not\models A \varphi_1 \]

\[ \varphi_2 = \Diamond \neg a \land \Box \neg P_1 \]
\[ s_3, s_4 \models A \varphi_2 \]
\[ s_1, s_2 \not\models A \varphi_2 \]
\( \varphi = AX A(\Diamond \neg a \land \Box \neg AF(Xb \land \neg a))) \)

\( \varphi_1 = F(Xb \land \neg a) \)

\( s_2 \models A \varphi_1 \)

\( s_1, s_3, s_4 \not\models A \varphi_1 \)

\( \varphi_2 = \Diamond \neg a \land \Box \neg P_1 \)

\( s_3, s_4 \models A \varphi_2 \)

\( s_1, s_2 \not\models A \varphi_2 \)
\( \varphi = AX A(\neg F a \land \neg G a F (X b \land \neg a)) \)

\( \varphi_1 = F (X b \land \neg a) \)
\[ s_2 \models A \varphi_1 \]
\[ s_1, s_3, s_4 \not\models A \varphi_1 \]

\( \varphi_2 = \neg F a \land \neg G P_1 \)
\[ s_3, s_4 \models A \varphi_2 \]
\[ s_1, s_2 \not\models A \varphi_2 \]

\( \varphi_3 = X P_2 \)
Model checking $B(L)$ [Emerson & Lei, 1987]

\[
\varphi = AX A(\neg a \land \neg AF(Xb \land \neg a))
\]

\[
\varphi_1 = F(Xb \land \neg a)
\]

\[
s_2 \models A \varphi_1
\]

\[
s_1, s_3, s_4 \not\models A \varphi_1
\]

\[
\varphi_2 = \neg a \land \neg P_1
\]

\[
s_3, s_4 \models A \varphi_2
\]

\[
s_1, s_2 \not\models A \varphi_2
\]

\[
\varphi_3 = XP_2
\]

\[
s_3, s_4 \models A \varphi_3
\]

\[
s_1, s_2 \not\models A \varphi_3
\]
Model checking $B(L)$ [Emerson & Lei, 1987]

\[ \varphi = \text{AX} \, \text{A} (\Box \neg a \land 
\Diamond \neg \text{AF} (Xb \land \neg a)) \]

\[ \varphi_1 = \text{F} (Xb \land \neg a) \]

\[ s_2 \models \text{A} \varphi_1 \]

\[ s_1, s_3, s_4 \not\models \text{A} \varphi_1 \]

\[ \varphi_2 = \Box \neg a \land \Diamond \neg P_1 \]

\[ s_3, s_4 \models \text{A} \varphi_2 \]

\[ s_1, s_2 \not\models \text{A} \varphi_2 \]

\[ \varphi_3 = X P_2 \]

\[ s_3, s_4 \models \text{A} \varphi_3 \]

\[ s_1, s_2 \not\models \text{A} \varphi_3 \]

\[ s_3, s_4 \models \varphi \]

\[ s_1, s_2 \not\models \varphi \]
**Upper bounds**

**Fact.** If model checking for $L$ is in $C$, then model checking for $B(L)$ is in $P^C$.

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

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 $C$, then model checking for $B(L)$ is in $P^C$.

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

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 $C$, then model checking for $B(L)$ is in $P^C$.

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

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 $L_0 \defeq \{X P, P \lor P'\}$ 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?

\[ \text{CTL} \rightarrow \text{ECTL} \]

\[ E_a U b \quad E^\infty F_a \]

Model Checking Branching-Time Logics – p.9/28
What expressivity/complexity compromise?

\[ A(\Box a \land \Box b \Rightarrow c \lor d) \]

\[ ECTL^+ \]

\[ E a \lor b \quad EF a \]

\[ CTL \rightarrow ECTL \]
What expressivity/complexity compromise?

\[ E(a \cup a' \cup b') \quad A(\exists a \land \exists b \Rightarrow c \cup d) \]

\[ CTL^+ \rightarrow ECTL^+ \]

\[ CTL \rightarrow ECTL \]

\[ E(a \cup b) \quad E\exists \neg a \]

Model Checking Branching-Time Logics – p.9/28
What expressivity/complexity compromise?

\[ E(a \cup a' \cup b') \quad A(\Diamond a \land \Diamond b \Rightarrow c \cup d) \]

\[ \text{CTL}^+ \rightarrow \text{ECTL}^+ \]

\[ \text{CTL} \rightarrow \text{ECTL} \]

\[ \text{ECTL} \]

\[ \text{FCTL} \]

\[ A \Diamond a \Rightarrow \Diamond b / F c \]

\[ \text{PSPACE}-\text{comp.} \quad \text{NP}-\text{hard and coNP}-\text{hard} \]
What expressivity/complexity compromise?

\[ E(a \cup a' \cup b') \quad A(\Box a \land \Box b \Rightarrow c \cup d) \]

\[ CTL^+ \rightarrow ECTL^+ \]

\[ CTL \rightarrow ECTL \rightarrow GFCTL \]

\[ FCTL \]

\[ A_\infty \Box \exists_\infty F_a \Rightarrow \infty F_c \]

\[ A_\infty F a \Rightarrow \infty F b \]

\[ E a \cup b \]

\[ E \Box F a \]

\[ A_\infty E \exists_\infty F_a \cup b U c \]
What expressivity/complexity compromise?

\[
\begin{align*}
\text{CTL}^+ & \rightarrow \text{ECTL}^+ \rightarrow \text{CTL}^* \\
\text{CTL} & \rightarrow \text{ECTL} \rightarrow \text{GFCTL} \\
\text{FCTL} & \rightarrow \text{GFCTL} \\
\end{align*}
\]
What expressivity/complexity compromise?

CTL

ECTL

GFCTL

FCTL

CTL *

PSPACE-comp.

P-hard and coNP-hard

E(aUa \land a'Ub')

A(\neg a \land \neg b \Rightarrow c \land d)

E\neg (a \Rightarrow Xb)

A_{\infty}\exists_{\infty}Fa \land \exists_{\infty}Fc \land bUc

A_{\infty}Fa \Rightarrow F_{\infty}Fc

E_{\infty}Fa

NP-hard and coNP-hard

PSPACE-compact.
Let $\mathcal{I}$ be a 3SAT instance. E.g.

$\mathcal{I}$ is $\left( x_1 \lor \overline{x_2} \lor \overline{x_4} \right) \land \left( \overline{x_1} \lor \cdots \right) \land \cdots$
NP-complete path modalities

Let $\mathcal{I}$ be a 3SAT instance. E.g. $\mathcal{I}$ is $(x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (\overline{x_1} \lor \cdots) \land \cdots$
Let $\mathcal{I}$ be a 3SAT instance. E.g. $\mathcal{I}$ is $(x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (\overline{x_1} \lor \cdots) \land \cdots$.

$\mathcal{I}$ is satisfiable iff $q_0 \models E \left[ (Fx_1 \lor F\overline{x_2} \lor F\overline{x_4}) \land (\overline{Fx_1} \lor \cdots) \land \cdots \right]$
NP-complete path modalities

Let $\mathcal{I}$ be a 3SAT instance. E.g. $\mathcal{I}$ is $(x_1 \lor \overline{x}_2 \lor \overline{x}_4) \land (\overline{x}_1 \lor \cdots) \land \cdots$.

$I$ is satisfiable iff $q_0 \models E \left[ (Fx_1 \lor F\overline{x}_2 \lor F\overline{x}_4) \land (F\overline{x}_1 \lor \cdots) \land \cdots \right]$

iff $q_0 \models E \left[ (Xx_1 \lor XX\overline{x}_2 \lor XXX\overline{x}_4) \land (X\overline{x}_1 \lor \cdots) \land \cdots \right]$
NP-complete path modalities

Let $I$ be a 3SAT instance. E.g. $I$ is $(x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (\overline{x_1} \lor \cdots) \land \cdots$.

$I$ is satisfiable iff $q_0 \models E \left[ (Fx_1 \lor F\overline{x_2} \lor F\overline{x_4}) \land (\overline{Fx_1} \lor \cdots) \land \cdots \right]$

iff $q_0 \models E \left[ (Xx_1 \lor XX\overline{x_2} \lor XXXX\overline{x_4}) \land (\overline{Xx_1} \lor \cdots) \land \cdots \right]$

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 $L^1(\infty, X, U)$ 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. $\Delta^p_2$, 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

\[ B^*(X) \]

\[ B^+(X) \]

\[ B(X) \quad B(F) \quad B(U) \quad B(U, X) \]
Towards a complete picture

$B^*(X) \rightarrow B^*(F) \rightarrow B^*(U) \rightarrow B^*(U, X)$

$B^+(X) \rightarrow B^+(F) \rightarrow B^+(U) \rightarrow B^+(U, X)$

$B(X) \rightarrow B(F) \rightarrow B(U) \rightarrow B(U, X)$
Towards a complete picture

NP- and coNP-hard

\[ B^*(X) \rightarrow B^*(F) \rightarrow B^*(U) \rightarrow B^*(U, X) \]

PSPACE-comp.

\[ \text{in } \Delta^p_2 \]

P-comp.

\[ B(X) \rightarrow B(F) \rightarrow B(U) \rightarrow B(U, X) \]
Towards a complete picture

\[ B^*(X) \rightarrow B^*(F) \rightarrow B^*(U) \rightarrow B^*(U, X) \]

NP- and coNP-hard

\[ B^+(X) \rightarrow B^+(F) \rightarrow B^+(U) \rightarrow B^+(U, X) \]

PSPACE-comp.

\[ B^*(F, X) \text{ in } \Delta^p_2 \]

GFCTL

\[ FCTL \]

P-comp.

\[ B(X) \rightarrow B(F) \rightarrow B(U) \rightarrow B(U, X) \]

NP- and coNP-hard

\[ B^+(F) \rightarrow B^+(U) \rightarrow B^+(U, X) \]

PSPACE-comp.
\( \Delta^p_2 \) and the polynomial-time hierarchy

\[
\begin{align*}
\Pi^p_2 & \quad \Sigma^p_2 = \text{NP}^\text{NP} = \text{NP}^\text{coNP} & \text{Q}_{\Sigma_2^p} \text{SAT} \\
\Delta^p_2 & \quad = \text{P}^\text{NP} = \text{P}^\text{coNP} & \text{SAT-SLP} \\
\Theta^p_2 & \quad = \text{P}^\text{NP[log]} = \text{P}^\text{NP} & \text{PARITY-SAT} \\
\Sigma^p_1 \cap \Pi^p_1 & \quad = \text{DP} & \text{SAT-UNSAT} \\
\Pi^p_1 & \quad \Sigma^p_1 = \text{NP} & \text{SAT}
\end{align*}
\]
A SAT-SLP is a sequence of Boolean assignments that use adaptive SAT problems:

\[
\begin{align*}
x_1 & : \iff \exists Z_1. B_1(Z_1) & \text{e.g. } \exists z_1, z_2, \bar{z}_3. [(z_1 \lor z_2 \lor \bar{z}_3) \land \cdots] \\
x_2 & : \iff \exists Z_2. B_2(x_1, Z_2) \\
x_3 & : \iff \exists Z_3. B_2(x_1, x_2, Z_3) & \text{e.g. } x_1 \land x_2 \\
\vdots & \\
x_n & : \iff \exists Z_n. B_n(x_1, x_2, \ldots, x_{n-1}, Z_n)
\end{align*}
\]

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:

\[ x_1 \iff \exists Z_1 B_1(Z_1) \quad \text{e.g. } \exists z_1, z_2, z_3. [(z_1 \lor z_2 \lor \bar{z}_3) \land \cdots] \]
\[ x_2 \iff \exists Z_2 B_2(x_1, Z_2) \]
\[ x_3 \iff \exists Z_3 B_2(x_1, x_2, Z_3) \quad \text{e.g. } x_1 \land x_2 \]
\[ \vdots \]
\[ x_n \iff \exists Z_n B_n(x_1, x_2, \ldots, x_{n-1}, Z_n) \]

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 \(\Delta^p_2\)-complete.
**NB:** This requires that queries are adaptive in depth and width.
Δ^p_2-hardness of \( B^+(F) \) model checking – 1

\[
\begin{align*}
x_1 & : \iff \exists Z. B_1(Z) \\
x_2 & : \iff \exists Z. B_2(x_1, Z) \\
x_3 & : \iff \exists Z. B_3(x_1, x_2, Z) \\
& \quad \vdots \\
x_n & : \iff \exists Z. B_n(x_1, \ldots, x_{n-1}, Z)
\end{align*}
\]
\[\Delta_2^p\text{-hardness of } B^+(F) \text{ model checking – 1}\]

\[
x_1 :\iff \exists Z.B_1(Z)
\]
\[
x_2 :\iff \exists Z.B_2(x_1, Z)
\]
\[
x_3 :\iff \exists Z.B_3(x_1, x_2, Z)
\]
\[
\vdots
\]
\[
x_n :\iff \exists Z.B_n(x_1, \ldots, x_{n-1}, Z)
\]
\( \Delta_{2}^{p} \)-hardness of \( B^+(F) \) model checking – 1

\[
x_1 :\Leftrightarrow \exists Z. B_1(Z) \\
x_2 :\Leftrightarrow \exists Z. B_2(x_1, Z) \\
x_3 :\Leftrightarrow \exists Z. B_3(x_1, x_2, Z) \\
\vdots \\
x_n :\Leftrightarrow \exists Z. B_n(x_1, \ldots, x_{n-1}, Z)
\]

\[
x_1 = T \iff q_0 \models E B_1(Fz_1, \ldots, Fz_p) \\
\text{“} q_0 \models \varphi_1 \text{”}
\]
$\Delta^p_2$-hardness of $B^+(F)$ model checking – 1

\[
x_1 :\iff \exists Z. B_1(Z)
\]
\[
x_2 :\iff \exists Z. B_2(x_1, Z)
\]
\[
x_3 :\iff \exists Z. B_3(x_1, x_2, Z)
\]
\[
\vdots
\]
\[
x_n :\iff \exists Z. B_n(x_1, \ldots, x_{n-1}, Z)
\]

\[
x_1 = T \iff q_0 \models E B_1(Fz_1, \ldots, Fz_p)
\]

\[
x_2 = T \iff q_0 \models E \left( B_2(Fx_1, Fz_1, \ldots, Fz_p) \wedge (Fx_1 \iff \varphi_1) \right)
\]

"$q_0 \models \varphi_1$"

"$q_0 \models \varphi_2$"
\[ x_1 \iff \exists Z. B_1(Z) \]

\[ x_2 \iff \exists Z. B_2(x_1, Z) \]

\[ x_3 \iff \exists Z. B_3(x_1, x_2, Z) \]

\[ \vdots \]

\[ x_n \iff \exists Z. B_n(x_1, \ldots, x_{n-1}, Z) \]

\[ x_1 = T \text{ iff } q_0 \models E \mathcal{B}_1(FZ_1, \ldots, FZ_p) \]

\[ x_2 = T \text{ iff } q_0 \models E \left[ \mathcal{B}_2(Fx_1, FZ_1, \ldots, FZ_p) \right. \]

\[ \quad \quad \quad \left. \wedge (Fx_1 \iff \varphi_1) \right] \]

\[ x_n = T \text{ iff } q_0 \models E \left[ \mathcal{B}_n(Fx_1, \ldots, Fx_{n-1}, FZ_1, \ldots, FZ_p) \right. \]

\[ \quad \quad \quad \left. \wedge (Fx_1 \iff \varphi_1) \wedge \cdots \wedge (Fx_{n-1} \iff \varphi_{n-1}) \right] \]
$\Delta^p_2$-hardness of $B^+(F)$ model checking – 1

$x_1 :\iff \exists Z.B_1(Z)$

$x_2 :\iff \exists Z.B_2(x_1, Z)$

$x_3 :\iff \exists Z.B_3(x_1, x_2, Z)$

\[ \vdots \]

$x_n :\iff \exists Z.B_n(x_1, \ldots, x_{n-1}, Z)$

\[ x_1 = T \text{ iff } q_0 \models E B_1(Fz_1, \ldots, Fz_p) \quad \text{“}q_0 \models \varphi_1\text{”} \]

\[ x_2 = T \text{ iff } q_0 \models E \left[ B_2(Fx_1, Fz_1, \ldots, Fz_p) \right. \]
\[ \left. \quad \land (Fx_1 \iff \varphi_1) \right] \quad \text{“}q_0 \models \varphi_2\text{”} \]

\[ \vdots \]

\[ x_n = T \text{ iff } q_0 \models E \left[ B_n(Fx_1, \ldots, Fx_{n-1}, Fz_1, \ldots, Fz_p) \right. \]
\[ \left. \quad \land (Fx_1 \iff \varphi_1) \land \cdots \land (Fx_{n-1} \iff \varphi_{n-1}) \right] \quad \text{“}q_0 \models \varphi_n\text{”} \]

Pb. $\varphi_n$ has exponential size!
\(\Delta^p_{2}\)-hardness of \(B^+(F)\) model checking – 2
\( \Delta_2^p \)-hardness of \( B^+(F) \) model checking – 2

\[
\varphi = E \left[ \begin{array}{c}
G \neg C \\
\land \land_{i=1}^n \left[ Fx_i \Rightarrow B_i(Fx_1, \ldots, Fx_{i-1}, Fz_1, \ldots, Fz_p) \right] \\
\land G[ (x_1 \lor \cdots \lor x_n) \Rightarrow EX(C \land EX \neg \varphi) ]
\end{array} \right]
\]
$\Delta^p_2$-hardness of $B^+(F)$ model checking – 2

\[
\varphi = E \left[ \begin{array}{c}
G \neg C \\
\land \land_{i=1}^{n}[Fx_i \Rightarrow B_i(Fx_1, \ldots, Fx_{i-1}, Fz_1, \ldots, Fz_p)] \\
\land G[(x_1 \lor \cdots \lor x_n) \Rightarrow EX(C \land EX \neg \varphi)]
\end{array} \right]
\]

Unfold the definition with $\varphi_0 = T$ and $\varphi_{i+1} = E[\cdots \land G[\cdots EX \neg \varphi_i]].$
\( \Delta^p_2 \)-hardness of \( B^+(F) \) model checking – 2

\[
\varphi = E \left[ G \neg C \land \bigwedge_{i=1}^{n} [Fx_i \Rightarrow B_i(Fx_1, \ldots, Fx_{i-1}, Fz_1, \ldots, Fz_p)] \land G[(\overline{x_1} \lor \cdots \lor \overline{x_n}) \Rightarrow \text{EX}(C \land \text{EX}\neg \varphi)] \right]
\]

Unfold the definition with \( \varphi_0 = T \) and \( \varphi_{i+1} = E[\cdots \land G[\cdots \text{EX}\neg \varphi_i]] \).

**Fact.** \( x_n = T \) in the SAT-SLP iff \( x_n \models \varphi_{2n-1} \) in the Kripke structure. Hence \( \Delta^p_2 \)-hardness of \( B^+(F) \) model checking.

(Similar tricks show \( \Delta^p_2 \)-hardness of \( B^+(\overline{F}) \) model checking.)
An almost complete picture

NP- and coNP-hard

$B^*(X) \rightarrow B^*(F) \rightarrow B^*(U) \rightarrow B^*(U, X)$

$B^+(X) \rightarrow B^+(F) \rightarrow B^+(U) \rightarrow B^+(U, X)$

PSPACE-comp.

$B^*(F, X)$

$B^+(F, X)$

$\Delta^p_2$-comp.

GFCTL

FCTL

P-comp.

$B(X) \rightarrow B(F) \rightarrow B(U) \rightarrow B(U, X)$
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 $\Delta^p_2$-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

\( \exists Z. B_2(y_1, y_2, Z) \)  
\( \exists Z. B_3(y'_1, y'_2, Z) \)

\( \exists Z. B_1(x_1, x_2, x_3, Z) \)

Fact: DAG-SAT is \( \mathsf{PSPACE} \)-complete

NB: TREE-SAT is \( \mathsf{PSPACE} \)-complete

[Gotlob 1995]
Circuits with SAT queries

Fact: DAG-SAT is $\Delta^p_2$-complete
Circuits with SAT queries

\[
\exists Z. B_2(y_1, y_2, Z) \\
\exists Z. B_3(y'_1, y'_2, Z) \\
\exists Z. B_1(x_1, x_2, x_3, Z)
\]

Fact: DAG-SAT is $\Delta^p_2$-complete
NB: TREE-SAT is $\Theta^p_2$-complete [Gottlob 1995]
Model checking for $B(L)$ translates into a tree of “blocks” containing parallel queries to a model checker for $L$. 

$$T :$$

$z_1, \ldots, z_k$

$B_1$

$B_2$

$B_4$

$B_5$

$B_6$

$B_7$

$y_1^1, \ldots, y_k^1$

$y_1^2, \ldots, y_k^2$

$y_1^3, \ldots, y_k^3$

$y_1^4, \ldots, y_k^4$
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 $\Delta^p_2$-complete
$B^*(X)$ translates into type 1xM queries

\[ q_i \models E(X((Xa) \Rightarrow (XXXb)) \lor XX \neg c) \]
\[ q_i \models E(X^2a \Rightarrow X^4b \lor \neg X^2c) \]
\[ z_i = T \text{ iff } q_i \models EB(X^{n_1}\psi_1, \ldots, X^{n_m}\psi_m) \]
$B^*(X)$ translates into type $1xM$ queries

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

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

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

Model Checking Branching-Time Logics – p.24/28
$B^*(X)$ translates into type $1 \times M$ queries

\[ q_i \models E(X((Xa) \Rightarrow (X Xb)) \lor X \neg c) \]
\[ q_i \models E(X^2a \Rightarrow X^4b \lor \neg X^2c) \]
\[ z_i = T \iff q_i \models EB(X^{n_1} \psi_1, \ldots, X^{n_m} \psi_m) \]

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

\[ \exists l_1, \ldots, l_m, Z'. B'_i(y^1_{l_1}, \ldots, y^m_{l_m}, l_1, \ldots, l_m, Z') \]
Flattening trees with type 1xM queries

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

Model Checking Branching-Time Logics – p.25/28
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 \( \mathbb{P}^{\mathbb{NP}[\log^{k+1}]} = \mathbb{P}^{\mathbb{NP}[\log^k]} \) [Castro & Seara 1996] we obtain:
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 $P^{NP[\log^{k+1}]} = P^{NP[\log^k]}$ [Castro & Seara 1996] we obtain:

**Theo.** Model checking $B^*(X)$ is in $P^{NP[\log^2]}$. 
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 $P^{NP[\log^{k+1}]} = P^{NP[\log^k]}$ [Castro & Seara 1996] we obtain:

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

**Theo.** It is in fact complete for $P^{NP[\log^2]}$. 
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 $\mathcal{E} a U_{=80}(AF_{\leq 35}b)$.

**Theo:** [Laroussinie, Markey, S. FoSSaCS’2002, S. 2003]
- Model checking *TCTL* on graphs with durations is $\Delta^p_2$-complete [LMS 02]
- Model checking the $B(F)$ fragment is $\Theta^p_2$-complete [Sch 03]