Home Assignment 1: Dynamic Logics

To hand in before or on November 13, 2013.

Electronic versions can be sent by email to dstan@lsv.ens-cachan.fr, paper versions should be handed in on the 13th or put in my mailbox at LSV, ENS Cachan.

The numbers in the margins next to exercises are indications of time and difficulty.

Syntax. A PDL formula is defined by the abstract syntax

\[ \varphi ::= \top | \bot | p | \neg \varphi | \varphi \land \varphi | \varphi \lor \varphi | \langle \alpha \rangle \varphi | [\alpha] \varphi \quad \text{(state formulæ)} \]

\[ \alpha ::= \varphi ? | \rightarrow | \alpha \cdot \alpha | \alpha + \alpha | \alpha^* \quad \text{(program formulæ)} \]

Semantics. Given a Kripke structure \( \mathcal{M} = \langle S, T, I, \alpha, \ell \rangle \) —where as in the lectures \( S \) is a set of states, \( T \subseteq S \times S \) is a transition relation, \( I \subseteq S \) is a set of initial states, and \( \ell \) is a labeling function from \( S \) to \( 2^{AP} \), we define inductively for each \( s \in S \),

\[ \mathcal{M}, s \models p \iff p \in \ell(s) \]

\[ \mathcal{M}, s \models \varphi \iff \mathcal{M}, s \models \varphi^2 \quad \text{iff} \quad s \xrightarrow{\varphi} s \]

\[ \mathcal{M}, s \models [\alpha] \varphi \iff \forall t \in S \quad s \xrightarrow{\alpha} t \Rightarrow \mathcal{M}, t \models \varphi \]

\[ \mathcal{M}, s \models \langle \alpha \rangle \varphi \iff \exists t \in S \quad s \xrightarrow{\alpha} t \land \mathcal{M}, t \models \varphi \]

where \( \alpha^n = \alpha \cdot \ldots \cdot \alpha \), \( n \)-times.

Intuitively, a PDL formula allows us to quantify over sets of possible runs of a form given by a regular expression. Moreover, we allow in these regular expressions testing an embedded state formula. Note that such a test does not trigger any transition in the underlying Kripke structure, and that the only symbol triggering a transition is the "next" symbol \( \rightarrow \).
Exercise 1. We define for $\varphi \in PDL$, $\mathcal{M}_1 = \{s \in S \mid \mathcal{M}, s \models \varphi\}$ as the set of states satisfying the formula.

Compute the following sets for the Kripke structure $\mathcal{M}_1$:

1. $\mathcal{J}_1[\rightarrow^* p \rightarrow 3]p$
2. $\mathcal{J}_1[\langle \neg p \rightarrow \rangle^*[\rightarrow^*]p]$, $\mathcal{J}_1[\langle \rightarrow p \rangle^*[\rightarrow^*]\neg p]$.

Exercise 2 (Identities).

1. Prove that $\langle \alpha \rangle \varphi \equiv \neg[\alpha] \neg \varphi$

2. Show that one can rewrite any formula $\varphi$ into an equivalent formula $\text{pnf}(\varphi)$ of linear size in the following grammar:

$$\varphi ::= \top | \bot | p | \neg p | \varphi \land \varphi | \varphi \lor \varphi | \langle \alpha \rangle \varphi | [\alpha] \varphi$$

$$\alpha ::= \varphi ? | \rightarrow \alpha \cdot \alpha | \alpha + \alpha | \alpha^*$$

3. Show the following equivalences:

$$[\alpha_1 \cdot \alpha_2] \varphi \equiv [\alpha_1][\alpha_2] \varphi$$
$$[\alpha_1 + \alpha_2] \varphi \equiv [\alpha_1] \varphi \land [\alpha_2] \varphi$$
$$[\varphi?] \psi \equiv \varphi \Rightarrow \psi$$

2 CTL equivalent fragments

Exercise 3.

1. Show that any formula $\varphi \in CTL$ can be rewritten in an equivalent formula $\tilde{\varphi} \in PDL$. What is the size of this new formula?

2. Suppose we restrict program formulæ to the following grammar:

$$\alpha ::= \varphi ? | \alpha \cdot \alpha | \alpha + \alpha | \rightarrow^*$$

show that the resulting state formulæ can be converted to equivalent CTL(EF) formulæ.
3. Suppose we allow again $\rightarrow$ in the grammar, show that we get $\text{CTL}(\text{EF}, \text{EX})$ expressivity.

### 3 PDL Model checking

**Exercise 4.** We focus now on the model checking problem of a Kripke structure against a formula.

1. Let $\phi = [\alpha]\psi$ a fixed formula. We write $\text{test}(\alpha) = \{\phi_1 \ldots \phi_k\}$ the set of formulæ that are tested (appearing as $\phi_i$?) in the program $\alpha$. Assume we have computed $[\psi]$ and $[[\phi_1] \ldots [\phi_k]]$, design an algorithm to compute $[\phi]$.

2. Adapt the CTL model checking algorithm to PDL formulæ.

3. Conclude on the complexity of PDL model checking.

### 4 LDL Model checking

We now focus on the linear dynamic logic. The syntax of LDL is the same as for PDL but the semantics is now defined on infinite words over $2^{AP} = \Sigma$. For $\sigma \in \Sigma^\omega$,

$$
\begin{align*}
\sigma, i \models p & \iff p \in \sigma_i \\
\vdots & \\
\sigma, i \models [\alpha] \phi & \iff \forall j \in \mathbb{N} \; i \xrightarrow{\alpha} j \Rightarrow \sigma, j \models \phi \\
\sigma, i \models \langle \alpha \rangle \phi & \iff \exists j \in \mathbb{N} \; i \xrightarrow{\alpha^1} t \wedge t \xrightarrow{\alpha^2} j \\
\sigma, i \models \alpha^* & \iff \exists n \geq 0 \; i \xrightarrow{\alpha^n} j
\end{align*}
$$

The model checking problem of a Kripke structure $\mathcal{M}$ against a formula $\phi$ consists in deciding whether $\mathcal{M} \models \exists \varphi$, that is to say whether there exists a run $\sigma \in \mathcal{I} \Sigma^\omega$ of $\mathcal{M}$ such that $\sigma, 0 \models \varphi$.

**Exercise 5** (Examples).

1. Let $AP = \{p\}$ and $\Sigma = 2^{AP}$. Give a LDL formula recognizing exactly the language $(\Sigma \{p\})^\omega$.

2. Consider the first example $\mathcal{M}_1$. Does $\mathcal{M}_1 \models [\rightarrow^* p?] \rightarrow^3 p$ hold? (Note the difference with PDL)

**Exercise 6** (Alternating Büchi automata). For any finite set $Q$, we denote by $\mathbb{B}^+(Q)$ the non-empty Boolean combinations of elements of $Q$, e.g., $p \wedge (q \vee r)$. For any $P \subseteq Q$, we write $P \models \xi$ iff $P$ satisfies $\xi$.

An (generalized) alternating Büchi automaton is a tuple $A = (Q, \Sigma, \delta, I, F, (R_i)_{1 \leq i \leq m})$ where

- $Q$ is a finite set of states
• \( \Sigma \) is a finite alphabet
• \( I \in \mathbb{B}^+ \) is the alternating initial condition
• \( \delta : Q \times \Sigma \to \mathbb{B}^+(Q) \) is the alternating transition function
• \( R_i \subseteq Q \) are the subsets of repeated states

A run of \( \mathcal{A} \) over the word \( w = a_0 \ldots \in \Sigma^\omega \) is a \( Q \)-labeled forest \((V, E, \rho)\) with \( E \subseteq V \times V \) and \( \rho : V \to Q \) such that

- \( \rho(\{ z \mid E^{-1}(z) = \emptyset \}) \models I \) (the set of roots satisfies the initial condition)
- for every \( x \in V \cap E^n(z) \) (every node of depth \( n \) from root \( z \)), \( \rho(E(x)) \neq \emptyset \) and \( \rho(E(x)) \models \delta(\rho(x), a_n) \). (That is to say, \( x \) is not a leaf and the set of children satisfies the alternating transition relation.)

A run is said to be accepting if each infinite branch \( b = x_0x_1 \ldots \) visits each \( R_i \) infinitely often.

We are interested here in the construction of an alternating Büchi automaton recognizing exactly the language accepted by a formula \( \varphi \in \text{LDL} \). For example, the previous formula \([\rightarrow^* p? \rightarrow^3]p\) is recognized by the following alternating Büchi automaton (notice that a classical Büchi automaton has only disjunctive transitive conditions):

\[
\begin{array}{c}
\emptyset \\
\downarrow \\
s_0 \{p\} \\
\downarrow \\
s_1 \Sigma \\
\downarrow \\
s_p \\
\downarrow \\
\Sigma \\
\downarrow \\
\{p\}
\end{array}
\]

\( \delta(s_0, \{p\}) = s_1 \land s_0 \)
\( \delta(s_0, \emptyset) = s_0 \)

In the following, we consider formulæ in pnf form.

1. Let \( \varphi = (\alpha)\psi \) a fixed formula in pnf form and \( \text{test}(\alpha) = \{ \varphi_1 \ldots \varphi_k \} \). Assume we have computed alternating Büchi automata \( \mathcal{A}_1 \ldots \mathcal{A}_k \) and \( \mathcal{A}_\psi \) recognizing respectively the languages of \( \varphi_1 \ldots \varphi_k, \psi \). Construct an alternating Büchi automaton for \( \varphi \).

**Hint:** Consider a finite automaton over the alphabet \( \{\rightarrow, \varphi_1 \ldots \varphi_k\} \) recognizing the regular expression \( \alpha \).

2. Complete the proof.