Combining forward and backward analyses of temporal properties

Damien Massé

LIX, École Polytechnique, Palaiseau, France,
masse@lix.polytechnique.fr,
http://www.lix.polytechnique.fr/~masse/

Abstract. In this paper we extend the well-known combination of forward and backward static analyses in abstract interpretation for the verification of complex temporal properties for transition systems. First, we show that this combination, whose results are often better than those obtained by using both analyses separately, can be used to check simple temporal properties with just one fixpoint. Then we extend this result to more complex temporal properties, including a superset of CTL in the case of non-game properties, and a superset of ATL in the case of game properties.

1 Introduction

Abstract interpretation [4, 7, 9] is a formal method for inferring general properties of a program. When the program is described as a transition system, two kinds of analyses can be done: backward analysis and forward analysis. Forward analysis simulates program computations, whereas backward analysis simulates *reverse* computations. Both analyses can be combined to obtain much better results, since each analysis may reduce the loss of precision introduced by the other [4]. However, only restricted kinds of properties expressed in the μ-calculus as intersections of properties in the form of $\nu X.(p \wedge \Diamond X)$ and $\mu X.(p \vee \Diamond X)$ are used, including for user-provided assertions [2][1].

More complex temporal properties (such as CTL and ATL [1]) are commonly checked in the *model checking* approach [3]. In that case, *abstractions* are commonly used to solve state explosion problems. Anyway, model-checking tools use either backward or forward analyses [12], but usually do not combine them, since one analysis is enough for finite concrete systems, and reversible temporal logics [10] are not used for specifications (forward logics, that use only *predecessor* operators, are used instead)[2].

In this paper, we show that forward analysis can still be combined with backward analysis in many model-checking temporal specifications. We will study

[1] In fact, the implemented tool [2] can be used to prove the negation of these properties, not these properties.

[2] It is shown in [10] that state-based abstractions are not complete when checking reversible temporal specifications.

non-game properties (a subset of μ-calculus formulas), and game properties (a subset of $A\mu$ formulas [1]). This combination can lead to better results, especially when using widening and narrowing techniques [4] to deal with infinite abstract lattices.

2 Standard combination of backward and forward analysis

The combination of backward and forward analyses was originally introduced in Cousot's thesis [4] in order to approximate the intersection of backward and forward transition system collecting semantics. It is widely known and used in abstract interpretation, since it enables to combine information given by abstract backward and forward operators, and thus to reduce the effects of the loss of information due to the abstraction.

In this section we recall the results that justify the correctness of this combination.

2.1 Combination of fixpoints

Lemma 1. *Let $P^\flat(\sqsubseteq^\flat, \bot^\flat, \top^\flat, \sqcap^\flat, \sqcup^\flat)$ and $P^\sharp(\sqsubseteq^\sharp, \bot^\sharp, \top^\sharp, \sqcap^\sharp, \sqcup^\sharp)$ be complete lattices with a Galois connection $P^\flat \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} P^\sharp$, $F^\flat \in P^\flat \to P^\flat$ and $B^\flat \in P^\flat \to P^\flat$ be two monotonic functions, and let*

$$L^\flat = \mathrm{gfp}\, \lambda Z.(Z \sqcap^\flat F^\flat(Z) \sqcap^\flat B^\flat(Z))$$

If $F^\sharp \in P^\sharp \to P^\sharp$ and $B^\sharp \in P^\sharp \to P^\sharp$ are monotonic and satisfy $\alpha \circ F^\flat \circ \gamma \sqsubseteq^\sharp F^\sharp$ and $\alpha \circ B^\flat \circ \gamma \sqsubseteq^\sharp B^\sharp$, then the sequence $(X_n)_{n \in \mathcal{N}}$ defined by $X_0 = \top^\sharp$, $X_{2n+1} = X_{2n} \sqcap^\sharp F^\sharp(X_{2n})$, and $X_{2n+2} = X_{2n+1} \sqcap^\sharp B^\sharp(X_{2n+1}), \forall n \geq 0$ is such that:

$$\forall n \geq 0,\ \alpha(L^\flat) \sqsubseteq^\sharp X_{n+1} \sqsubseteq^\sharp X_n$$

The optimality of this approach has been proved in [8]: it has been shown that $L^\sharp = \mathrm{gfp}\, \lambda Z.(Z \sqcap^\sharp F^\sharp(Z) \sqcap^\sharp B^\sharp(Z))$ is the greatest lower bound of the set $E^\sharp$ defined inductively as:

- $\top^\sharp \in E^\sharp$
- If Z is in $E^\sharp$ then so are $F^\sharp(Z)$ and $B^\sharp(Z)$.
- If Z_1 and Z_2 are in $E^\sharp$ then so are $Z_1 \sqcap^\sharp Z_2$ and $Z_1 \sqcup^\sharp Z_2$.

Therefore, $L^\sharp$ is the best upper approximation of $\alpha(L^\flat)$ that can be obtained using $F^\sharp$ and $B^\sharp$.

2.2 Standard backward-forward combination

The standard backward-forward combination [4] derives from an application of this lemma to the particular case of backward and forward collecting semantics:

$F^\flat$, $B^\flat$, $F^\sharp$ and $B^\flat$ are instantiated as follows:

$$\begin{aligned}
F^\flat &= \lambda Y.\mathrm{lgfp}_1 \lambda X.(Y \sqcap^\flat f^\flat(X)) \\
B^\flat &= \lambda Y.\mathrm{lgfp}_2 \lambda X.(Y \sqcap^\flat b^\flat(X)) \\
F^\sharp &= \lambda Y.\mathrm{lgfp}_1 \lambda X.(Y \sqcap^\sharp f^\sharp(X)) \\
B^\sharp &= \lambda Y.\mathrm{lgfp}_2 \lambda X.(Y \sqcap^\sharp b^\sharp(X))
\end{aligned}$$

where lgfp means either lfp or gfp, and $f^\flat$, $b^\flat \in P^\flat \to P^\flat$, $f^\sharp$, $b^\sharp \in P^\sharp \to P^\sharp$ are monotonic. When $\alpha \circ f^\flat \circ \gamma \sqsubseteq^\sharp f^\sharp$ and $\alpha \circ b^\flat \circ \gamma \sqsubseteq^\sharp b^\sharp$, the conditions of Lemma 1 are satisfied [7].

Now, let $P^\flat = \wp(\Sigma)$, Σ a set of states[3], and $\tau \in \wp(\Sigma \times \Sigma)$ a transition relation. As usual, we define pre, $\widetilde{pre}$, $post$, $\widetilde{post}$ as:

$$\begin{aligned}
post(X) &= \{s' \mid \exists s : \langle s, s' \rangle \in \tau \wedge s \in X\} \\
\widetilde{post}(X) &= \{s' \mid \forall s : \langle s, s' \rangle \in \tau \Rightarrow s \in X\} \\
pre(X) &= \{s \mid \exists s' : \langle s, s' \rangle \in \tau \wedge s' \in X\} \\
\widetilde{pre}(X) &= \{s \mid \forall s' : \langle s, s' \rangle \in \tau \Rightarrow s' \in X\}
\end{aligned}$$

Given $\mathcal{I}$,$\mathcal{F} \subseteq \Sigma$, sets of initial and final states, $f^\flat = \lambda X.(\mathcal{I} \cup post(X))$ and $b^\flat = \lambda X.(\mathcal{F} \cup pre(X))$ (and $\mathrm{lgfp}_1 = \mathrm{lgfp}_2 = \mathrm{lfp}$), we have [4]:

$$L^\flat = F^\flat(\top^\flat) \sqcap^\flat B^\flat(\top^\flat) = \mathrm{lfp}\, f^\flat \cap \mathrm{lfp}\, b^\flat \tag{1}$$

By computing $\gamma(L^\sharp)$[4], we obtain a good upper approximation of $F^\flat(\Sigma) \cap B^\flat(\Sigma)$ (at least equal to $\gamma(F^\sharp(\top^\sharp) \sqcap^\sharp B^\sharp(\top^\sharp))$).

$B^\flat(\top^\flat)$ is the set of states satisfying the μ-calculus formula $\mu X.(F \vee \Diamond X)$, where F is satisfied by $\mathcal{F}$. Therefore the method is used to analyze reachability (rather unreachability, since we compute a superset of reachable states, or a subset of unreachable states) properties. Equation (1) holds with the μ-calculus formula $\nu X.(F \wedge \Diamond X)$ too, and this result allows the analysis of termination properties.

For other formulas (like $\mu X.(F \vee \Box X)$), equation (1) does not hold in general: a state may satisfy the backward property and be reachable from an initial state which does not satisfy the backward property (an example is given in Figure 1).

In the next section, we will prove that under certain conditions we can still use the combination for formulas with a single fixpoint.

3 Extension with a single fixpoint

As we want to deal with μ-calculus formulas in this section, we assume that $f^\flat = \lambda X.(\mathcal{I} \cup post(X))$ and that $\mathrm{lgfp}_1 = \mathrm{lfp}$. Computing (or approximating) $\alpha(L^\flat)$ is useless if we do not know $L^\flat$. Equation (1) must hold in order to approximate

[3] So $\top^\flat = \Sigma$, $\bot^\flat = \emptyset$, $\sqcap^\flat = \cap$, $\sqcup^\flat = \cup$.

[4] We may over-approximate the greatest fixpoint with a narrowing [7].

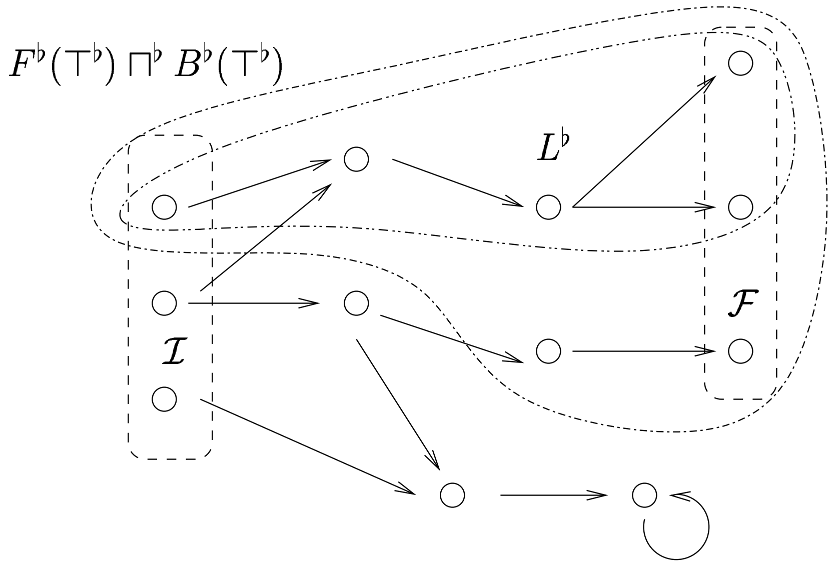

Fig. 1. Example where $L^\flat \neq F^\flat(\top^\flat) \sqcap^\flat B^\flat(\top^\flat)$. Here, $b^\flat = \lambda X.(\mathcal{F} \cup \widetilde{pre}(X))$ and $f^\flat = \lambda X.(\mathcal{I} \cup post(X))$. $L^\flat$ is then the set of states belonging to a trace of states which satisfy the backward property.

the sets of descendant states of $\mathcal{I}$ which satisfy $B^\flat(\top^\flat)$. Anyway, if we want to check a temporal formula, we just need to know the set of initial states which satisfy $B^\flat(\top^\flat)$, that is $\mathcal{I} \cap B^\flat(\top^\flat)$. Thus the combination is useful if the equality:

$$\mathcal{I} \cap B^\flat(\top^\flat) = \mathcal{I} \cap L^\flat \tag{2}$$

is satisfied[5].

Lemma 2. *Assuming that* $f^\flat = \lambda X.(\mathcal{I} \cup post(X))$*, if* $L^\flat = F^\flat(B^\flat(\Sigma))$*, then* $\mathcal{I} \cap B^\flat(\Sigma) = \mathcal{I} \cap L^\flat$.

Proof. As $L^\flat = F^\flat(B^\flat(\Sigma)) = \text{lfp}\ \lambda X.(B^\flat(\Sigma) \cap (f^\flat(X)))$, it is clear that $L^\flat \subseteq B^\flat(\Sigma)$. Moreover, the first iteration of the least fixpoint is $B^\flat(\Sigma) \cap f^\flat(\emptyset)$, which is equal to $B^\flat(\Sigma) \cap \mathcal{I}$. So we have $B^\flat(\Sigma) \cap \mathcal{I} \subseteq L^\flat$, which proves the equality.

With $B^\flat = \lambda Y.\text{lgfp}_2 \lambda X.(Y \cap b^\flat(X))$, we have the following lemma:

Lemma 3. *If* $\forall (X,Y) \in \wp(\Sigma)^2, Y \subseteq b^\flat(X) \Leftrightarrow Y \subseteq b^\flat(X \cap post(Y))$*, then the hypothesis of Lemma 2 holds. Thus, equation (2) holds.*

Proof. We note that the hypothesis implies:

$$\forall (X,Y) \in \wp(\Sigma)^2, Y \subseteq b^\flat(X) \Leftrightarrow Y \subseteq b^\flat(X \cap f^\flat(Y))$$

[5] And this equality is satisfied when equation (1) holds.

We want to prove that $L^\flat = F^\flat(B^\flat(\Sigma))$. Left inclusion is the consequence of the optimality of $L^\flat$:

$$\begin{aligned} L^\flat &= \operatorname{gfp} \lambda Z.(Z \cap F^\flat(Z) \cap B^\flat(Z)) \\ &\subseteq F^\flat(F^\flat(\Sigma) \cap B^\flat(\Sigma)) \cap B^\flat(F^\flat(\Sigma) \cap F^\flat(\Sigma)) \\ &\subseteq F^\flat(B^\flat(\Sigma)) \end{aligned}$$

Thus, to prove the equality, we need to check that $F^\flat(B^\flat(\Sigma))$ is a fixpoint of $\lambda Z.(Z \cap F^\flat(Z) \cap B^\flat(Z))$, that is, to prove that $F^\flat(B^\flat(\Sigma)) \subseteq F^\flat(F^\flat(B^\flat(\Sigma)))$ and $F^\flat(B^\flat(\Sigma)) \subseteq B^\flat(F^\flat(B^\flat(\Sigma)))$. The former inequality is true because $F^\flat \circ F^\flat = F^\flat$. To prove the latter, we define $\Omega = F^\flat(B^\flat(\Sigma))$ and $\Omega' = B^\flat(F^\flat(B^\flat(\Sigma)))$. We distinguish two cases:

- if $\mathrm{lgfp}_2 = \mathrm{lfp}$, let X_n, $n \geq 0$ be the (transfinite) iteration sequence starting from $\emptyset$ for $b^\flat$. We will prove that $\Omega \cap X_n \subseteq \Omega'$, for all $n \geq 0$. This is true if $n = 0$, because $X_0 = \emptyset$.
  If n is a successor ordinal, and if the inequality holds for $n-1$, we have $\Omega \cap X_n \subseteq b^\flat(X_{n-1})$. Using the hypothesis of the lemma, we obtain $\Omega \cap X_n \subseteq b^\flat(X_{n-1} \cap f^\flat(\Omega \cap X_n))$. By monotony, $f^\flat(\Omega \cap X_n) \subseteq f^\flat(\Omega)$. So, since $X_{n-1} \subseteq B^\flat(\Sigma)$:

$$\begin{aligned} X_{n-1} \cap f^\flat(\Omega \cap X_n) &\subseteq X_{n-1} \cap B^\flat(\Sigma) \cap f^\flat(\Omega) \\ &\subseteq X_{n-1} \cap \Omega \\ &\subseteq \Omega' \end{aligned}$$

  Hence $\Omega \cap X_n$ is included in $\Omega \cap b^\flat(\Omega')$, which is equal to Ω' by definition of Ω'.
  When n is a limit ordinal ($b^\flat$ may be not continuous), if $\Omega \cap X_{n'} \subseteq \Omega'$ for all $n' < n$, then $\Omega \cap X_n = \Omega \cap \bigcup_{n'<n} X_{n'} \subseteq \Omega'$.
  By transfinite induction, $\Omega \cap X_n \subseteq \Omega'$ for all n. As the upper bound of (X_n) is $B^\flat(\Sigma)$, which includes Ω, we have $\Omega \subseteq \Omega'$.
- if $\mathrm{lgfp}_2 = \mathrm{gfp}$, let X_n, $n \geq 0$ be the (transfinite) iteration sequence starting from Σ for $\lambda X.(\Omega \cap b^\flat)$. The limit of X_n is Ω'. $X_1 = \Omega \cap b^\flat(\Sigma) = \Omega$, since $\Omega \subseteq B^\flat(\Sigma) \subseteq b^\flat(\Sigma)$. Moreover, since $B^\flat(\Sigma) = b^\flat(B^\flat(\Sigma))$, $\Omega \subseteq b^\flat(B^\flat(\Sigma))$, so $\Omega \subseteq b^\flat(B^\flat(\Sigma) \cap f^\flat(\Omega))$. As $B^\flat(\Sigma) \cap f^\flat(\Omega) = \Omega$, we have $\Omega \subseteq b^\flat(\Omega)$, and $X_2 = \Omega$. Thus $X_n = \Omega$ for all $n \geq 1$, and $\Omega' = \Omega$.

Application: With $b^\flat = \lambda X.(A \cup (B \cap \mathit{pre}(X)) \cup (C \cap \widetilde{\mathit{pre}}(X)))$.
If $Y \subseteq b^\flat(X)$, then, $\forall y \in Y$:

- if $y \in A$, then $y \in b^\flat(X \cap \mathit{post}(Y))$.
- if $y \in B \cap \mathit{pre}(X)$, then $\exists x \in X$ such that $\langle y, x\rangle \in \tau$. Therefore, since $y \in Y$, $x \in \mathit{post}(Y)$, and $y \in B \cap \mathit{pre}(X \cup \mathit{post}(Y))$.
  Thus $y \in b^\flat(X \cap \mathit{post}(Y))$.
- if $y \in C \cap \widetilde{\mathit{pre}}(X)$, then $\forall x \in \Sigma$, $\langle y, x\rangle \in \tau \Rightarrow x \in X$. As $y \in Y$, $\langle y, x\rangle \in \tau \Rightarrow x \in \mathit{post}(Y)$, so $\forall x \in \Sigma$, $\langle y, x\rangle \in \tau \Rightarrow x \in X \cap \mathit{post}(Y)$.
  Thus $y \in C \cap \widetilde{\mathit{pre}}(X \cup \mathit{post}(Y))$, so $y \in b^\flat(X \cap \mathit{post}(Y))$.

Therefore, $Y \subseteq b^\flat(X) \Rightarrow Y \subseteq b^\flat(X \cap \mathit{post}(Y))$. The other side of the equivalence is automatic. Thus the hypothesis of Lemma 3 is satisfied, and equation (2) holds.

So we can use the backward-forward combination to enhance the verification of properties in the form of: $\sigma X.(A \vee (B \wedge \Diamond X) \vee (C \wedge \Box X))$ with $\sigma \in \{\mu, \nu\}$. These properties are interesting: they allow to distinguish between different kinds of non-determinism ("controllable" and "uncontrollable" non-determinism). We are not far from game properties, as we will see in section 5.

Unfortunately, the extension of this technique to the whole μ-calculus does not work: for example, a formula like $\mu X.(F \vee \Diamond\Diamond X)$ leaves "holes" in traces, preventing combination with forward analysis. However, it is possible, from this result, to extend it to a temporal logic expressive at least as CTL.

4 Extension to a larger specification language

In this section, we try to apply the backward-forward combination to the verification of some μ-calculus formulas. If φ is a formula, we denote by $[\![\varphi]\!]$ the set of states (in Σ) satisfying φ.

The formulas φ are defined by the grammar:

$$\varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box\varphi \mid \Diamond\varphi \mid \sigma X.(\varphi_1 \vee (\varphi_2 \wedge \Diamond X) \vee (\varphi_3 \wedge \Box X))$$

with $\sigma \in \{\mu, \nu\}$. It is worth noting that all these formulas are closed, and the defined temporal logic includes CTL. Obviously, the logic does not change if we replace $\sigma X.(\varphi_1 \vee (\varphi_2 \wedge \Diamond X) \vee (\varphi_3 \wedge \Box X))$ with $\sigma X.(\varphi_1 \wedge (\varphi_2 \vee \Diamond X) \wedge (\varphi_3 \vee \Box X))$ in the above grammar.

Our goal is to obtain a "good" upper approximation $\Omega_\varphi(\alpha(\mathcal{I}))$ of $\alpha(\mathcal{I} \cap [\![\varphi]\!])$, using backward-forward technique to enhance fixpoint computations. We assume that for all proposition p, we have an upper approximation $[\![p]\!]^\sharp$ of $\alpha([\![p]\!])$[6] (and an upper approximation $[\![\neg p]\!]^\sharp$ of $\alpha([\![\neg p]\!])$).

We need abstractions of *pre*, *post* and $\widetilde{\mathit{pre}}$, and we will respectively denote them by $\mathit{pre}^\sharp$,$\mathit{post}^\sharp$ and $\widetilde{\mathit{pre}}^\sharp$. Moreover we will denote by post^* and $\mathit{post}^{*\sharp}$ the functions $\lambda X.\mathrm{lfp}\ \lambda Y.(X \cup \mathit{post}(Y))$ and $\lambda X.\mathrm{lfp}\ \lambda Y.(X \sqcup^\sharp \mathit{post}^\sharp(Y))$ respectively. The following inequalities are assumed to be satisfied for all $X \subseteq \Sigma$:

$$\alpha \circ \mathit{pre}(X) \sqsubseteq^\sharp \mathit{pre}^\sharp \circ \alpha(X)$$
$$\alpha \circ \widetilde{\mathit{pre}}(X) \sqsubseteq^\sharp \widetilde{\mathit{pre}}^\sharp \circ \alpha(X)$$
$$\alpha \circ \mathit{post}(X) \sqsubseteq^\sharp \mathit{post}^\sharp \circ \alpha(X)$$
$$\alpha \circ \mathit{post}^*(X) \sqsubseteq^\sharp \mathit{post}^{*\sharp} \circ \alpha(X)$$

[6] which may be $\alpha([\![p]\!])$, if it is computable.

To simplify notations we denote by $L^{\natural}_{\text{lgfp}}(f,g)$, with $\natural \in \{\flat, \sharp\}$, the expression:

$$\text{gfp}\ \lambda Z.(\text{lfp}\ \lambda X.(Z \sqcap^{\natural} f(X)) \sqcap^{\natural} \text{lgfp}\ \lambda X.(Z \sqcap^{\natural} g(X)))$$

$L^{\natural}_{\text{lgfp}}(f,g)$ is the limit of the decreasing chain Z_n defined by $Z_0 = \top^{\natural}$, $Z_{2n+1} = Z_{2n} \sqcap^{\natural} \text{lfp}\ \lambda X.(X_{2n} \sqcap^{\natural} f(X))$ and $Z_{2n+2} = Z_{2n+1} \sqcap^{\natural} \text{lgfp}\ \lambda X.(X_{2n+1} \sqcap^{\natural} g(X))$.

If φ is a formula, we can now define : $\Omega_{\varphi} \in P^{\sharp} \to P^{\sharp}$ as follows:

$$\begin{aligned}
\Omega_{p}(S) &= [\![p]\!]^{\sharp} \sqcap^{\sharp} S \\
\Omega_{\varphi_1 \wedge \varphi_2}(S) &= \Omega_{\varphi_1}(S) \sqcap^{\sharp} \Omega_{\varphi_2}(S) \\
\Omega_{\varphi_1 \vee \varphi_2}(S) &= \Omega_{\varphi_1}(S) \sqcup^{\sharp} \Omega_{\varphi_2}(S) \\
\Omega_{\Box\varphi}(S) &= \widetilde{pre}^{\sharp}(\Omega_{\varphi}(post^{\sharp}(S))) \\
\Omega_{\Diamond\varphi}(S) &= pre^{\sharp}(\Omega_{\varphi}(post^{\sharp}(S))) \\
\Omega_{\sigma X.(\varphi_1 \vee (\varphi_2 \wedge \Diamond X) \vee (\varphi_3 \wedge \Box X))}(S) &= L^{\sharp}_{\text{lgfp}}(\ \lambda X.(S \sqcup^{\sharp} post^{\sharp}(X)), \\
&\qquad \lambda Y.(\ \Omega_{\varphi_1}(post^{*\sharp}(S)) \\
&\qquad\qquad \sqcup^{\sharp} \Omega_{\varphi_2}(post^{*\sharp}(S)) \sqcap^{\sharp} pre^{\sharp}(Y) \\
&\qquad\qquad \sqcup^{\sharp} \Omega_{\varphi_3}(post^{*\sharp}(S)) \sqcap^{\sharp} \widetilde{pre}^{\sharp}(Y)))
\end{aligned}$$

It is worth noting that even if we replace $post^{*\sharp}(S)$ by $\top^{\sharp}$ in the last line, we still have to compute it as the first iteration that leads to $L^{\sharp}_{\text{lgfp}}$. However, this replacement may not change the final result of Ω_{φ} and make the computation much faster (because the computation of $\Omega_{\varphi}(\top^{\sharp})$ can be simplified). The following theorem is valid with or without the replacement:

Theorem 1. *For any formula φ, and $\mathcal{I} \subseteq \Sigma$:*

$$\alpha(\mathcal{I} \cap [\![\varphi]\!]) \sqsubseteq^{\sharp} \alpha(\mathcal{I}) \sqcap^{\sharp} \Omega_{\varphi}(\alpha(\mathcal{I}))$$

Proof. The proof is by induction on the structure of φ. By monotony of α, it is obvious that $\alpha(\mathcal{I} \cap [\![\varphi]\!]) \sqsubseteq^{\sharp} \alpha(\mathcal{I})$, so we need to prove that $\alpha(\mathcal{I} \cap [\![\varphi]\!]) \sqsubseteq^{\sharp} \Omega_{\varphi}(\alpha(\mathcal{I}))$.

If $\varphi = p$, by monotony of α:

$$\alpha(\mathcal{I} \cap [\![p]\!]) \sqsubseteq^{\sharp} \alpha(\mathcal{I}) \sqcap^{\sharp} \alpha([\![p]\!]) \sqsubseteq^{\sharp} \alpha(\mathcal{I}) \sqcap^{\sharp} [\![p]\!]^{\sharp} \sqsubseteq^{\sharp} \Omega_{p}(\alpha(\mathcal{I}))$$

If $\varphi = \varphi_1 \wedge \varphi_2$:

$$\begin{aligned}
\alpha(\mathcal{I} \cap [\![\varphi]\!]) &= \alpha(\mathcal{I} \cap [\![\varphi_1]\!] \cap \mathcal{I} \cap [\![\varphi_2]\!]) \\
&\sqsubseteq^{\sharp} \alpha(\mathcal{I} \cap [\![\varphi_1]\!]) \sqcap^{\sharp} \alpha(\mathcal{I} \cap [\![\varphi_2]\!]) \\
&\sqsubseteq^{\sharp} \Omega_{\varphi_1}(\alpha(\mathcal{I})) \sqcap^{\sharp} \Omega_{\varphi_2}(\alpha(\mathcal{I})) \\
&\sqsubseteq^{\sharp} \Omega_{\varphi}(\alpha(\mathcal{I}))
\end{aligned}$$

If $\varphi = \varphi_1 \vee \varphi_2$, the calculus is quite the same, except that we use the fact that α is additive [7].

If $\varphi = \Box\varphi_1$, we have $\mathcal{I} \subseteq \widetilde{pre}(post(\mathcal{I}))$, so:

$$\begin{aligned}
\alpha(\mathcal{I} \cap [\![\varphi]\!]) &\sqsubseteq^\sharp \alpha(\widetilde{pre}(post(\mathcal{I}) \cap [\![\varphi_1]\!])) \\
&\sqsubseteq^\sharp \widetilde{pre}^\sharp(\alpha(post(\mathcal{I}) \cap [\![\varphi_1]\!])) \\
&\sqsubseteq^\sharp \widetilde{pre}^\sharp(\Omega_{\varphi_1}(\alpha \circ post(\mathcal{I}))) \\
&\sqsubseteq^\sharp \widetilde{pre}^\sharp(\Omega_{\varphi_1}(post^\sharp \circ \alpha(\mathcal{I}))) \\
&\sqsubseteq^\sharp \Omega_\varphi(\alpha(\mathcal{I}))
\end{aligned}$$

If $\varphi = \Diamond\varphi_1$, using the inequality $\mathcal{I} \cap pre([\![\varphi_1]\!]) \subseteq pre(post(\mathcal{I}) \cap [\![\varphi_1]\!])$, we can do the same calculus.
If $\varphi = \sigma X.(\varphi_1 \vee (\varphi_2 \wedge \Diamond X) \vee (\varphi_3 \wedge \Box X))$, let's define $h^\flat = \lambda X.([\![\varphi_1]\!] \cup [\![\varphi_2]\!] \cap \widetilde{pre}(X) \cup [\![\varphi_3]\!] \cap pre(X))$. Then $[\![\varphi]\!] = \text{lgfp}\, \lambda X.h^\flat(X)$.

We will use Lemma 1 with

$$\begin{aligned}
f^\flat &= \lambda X.(\mathcal{I} \cup post(X)) \\
f^\sharp &= \lambda X.(\alpha(\mathcal{I}) \sqcup^\sharp post^\sharp(X)) \\
b^\flat &= \lambda X.(post^*(\mathcal{I}) \cap ([\![\varphi_1]\!] \cup [\![\varphi_2]\!] \cap \widetilde{pre}(X) \cup [\![\varphi_3]\!] \cap pre(X))) \\
b^\sharp &= \lambda X.(\alpha(post^*(\mathcal{I}) \cap [\![\varphi_1]\!]) \\
&\quad \sqcup^\sharp \alpha(post^*(\mathcal{I}) \cap [\![\varphi_2]\!]) \sqcap^\sharp \widetilde{pre}^\sharp(X) \\
&\quad \sqcup^\sharp \alpha(post^*(\mathcal{I}) \cap [\![\varphi_3]\!]) \sqcap^\sharp pre^\sharp(X))
\end{aligned}$$

It is clear that $\alpha \circ b^\flat \circ \gamma \sqsubseteq^\sharp b^\sharp$ and $\alpha \circ f^\flat \circ \gamma \sqsubseteq^\sharp f^\sharp$ (given the standard properties that α is additive and $\alpha \circ \gamma$ is a lower closure operator [7]). Thus we have $\alpha(L^\flat_{\text{lgfp}}(f^\flat, b^\flat)) \sqsubseteq^\sharp L^\sharp_{\text{lgfp}}(f^\sharp, b^\sharp)$.

First, we prove that $\mathcal{I} \cap L^\flat_{\text{lgfp}}(f^\flat, b^\flat) = \mathcal{I} \cap [\![\varphi]\!]$. We have:

$$L^\flat_{\text{lgfp}}(f^\flat, h^\flat) \subseteq \text{lgfp}\, \lambda X.((\text{lfp}\, \lambda Y.f^\flat(Y)) \cap h^\flat(X)) \subseteq [\![\varphi]\!]$$

Applying Lemma 3 with $h^\flat$, we obtain $\mathcal{I} \cap [\![\varphi]\!] = \mathcal{I} \cap L^\flat_{\text{lgfp}}(f^\flat, h^\flat)$, so $\mathcal{I} \cap [\![\varphi]\!] = \mathcal{I} \cap \text{lgfp} \lambda X.((\text{lfp} \lambda Y.f^\flat(Y)) \cap h^\flat(X)) = \mathcal{I} \cap \text{lgfp} \lambda X.b^\flat(X)$. Then, applying Lemma 3 with $b^\flat$, we obtain: $\mathcal{I} \cap \text{lgfp}\, \lambda X.b^\flat(X) = \mathcal{I} \cap L^\flat_{\text{lgfp}}(f^\flat, b^\flat)$, proving the equality. So we proved:

$$\begin{aligned}
\alpha(\mathcal{I} \cap [\![\varphi]\!]) &= \alpha(\mathcal{I} \cap L^\flat_{\text{lgfp}}(f^\flat, b^\flat)) \\
&\sqsubseteq^\sharp \alpha(\mathcal{I}) \sqcap^\sharp \alpha(L^\flat_{\text{lgfp}}(f^\flat, b^\flat)) \\
&\sqsubseteq^\sharp \alpha(\mathcal{I}) \sqcap^\sharp L^\sharp_{\text{lgfp}}(f^\sharp, b^\sharp)
\end{aligned}$$

Now we need to check that:

$$L^\sharp_{\text{lgfp}}(f^\sharp, b^\sharp) \sqsubseteq^\sharp \Omega_\varphi(\alpha(\mathcal{I}))$$

By induction hypothesis, we see that $\Omega_\varphi(\alpha(\mathcal{I})) = L^\sharp_{\text{lgfp}}(f^\sharp, b'^\sharp)$ with $b'^\sharp$ satisfying $b^\sharp \sqsubseteq^\sharp b'^\sharp$, which complete the proof.

5 Extension to alternating-time temporal logic

Many properties on reactive systems are not easily expressible as μ-calculus formulas. This is true for game properties, which can be expressed as *alternating time μ-calculus* ($A\mu$) formulas [1], or as formulas of weaker game logics ATL and ATL*. In [13], basic abstract interpretation theory was applied on alternating transition systems, with a model-checking approach of abstraction. As we did with a subset of μ-calculus, we will try to apply forward-backward techniques to $A\mu$.

5.1 Alternating transition system, operators

An *alternating transition system* [1] is a tuple $\langle \Sigma, Q, \Delta, \Pi, \pi \rangle$, with Σ a set of *players*, Q a set of *states*, $\Delta = \{\delta_i : Q \to 2^{2^Q} \mid i \in \Sigma\}$ a set of *transition functions*, Π a set of propositions, and $\pi : \Pi \to 2^Q$ a function which associates each proposition to a set of states.

When the system is in state q, each player a must choose a set $Q_a \in \delta_a(q)$, and the successor of the state q must lie in $\bigcap_{a\in\Sigma} Q_a$ (it is assumed that the intersection is always a singleton, so the transition function is nonblocking and "deterministic"). Thus, if we want an equivalent of the *post* operator used in the non-game case, it would be:

$$Post(\sigma) = \bigcup_{q\in\sigma} \Big(\bigcap_{a\in\Sigma} \bigcup \delta_a(q) \Big)$$

As before, we can define $Post^*(\sigma) = \text{lfp}\, \lambda X.(\sigma \cup Post(X))$.

The equivalent of the *pre* and $\widetilde{pre}$ operators are the *controllable and uncontrollable predecessor relations*, defined in [13]. In general case, with $I \subset \Sigma \backslash \{\emptyset\}$, they are defined as:

$$q \in CPre_I(\sigma) \text{ iff } \exists(\tau_i \in \delta_i(q))_{i\in I}.\forall(\tau_i \in \delta_i(q))_{i\notin I}.\ \bigcap_{i\in\Sigma} \tau_i \subseteq \sigma$$
$$q \in UPre_I(\sigma) \text{ iff } \forall(\tau_i \in \delta_i(q))_{i\in I}.\exists(\tau_i \in \delta_i(q))_{i\notin I}.\ \bigcap_{i\in\Sigma} \tau_i \subseteq \sigma$$

$q \in CPre_I(\sigma)$ means that, when the system is in state q, the team I can force the successor state of q to be in σ. If $q \in UPre_I(\sigma)$, it means in state q, the team I cannot force the game outside σ. Of course, if there is only one player, these two operators are equivalent to *pre* and $\widetilde{pre}$.

5.2 Alternating-time μ-calculus

$A\mu$ formulas are generated by the grammar:

$$\varphi ::= p \mid \neg p \mid x \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle\!\langle I \rangle\!\rangle \bigcirc \varphi_1 \mid [\![I]\!] \bigcirc \varphi_1 \mid (\mu x.\varphi_1) \mid (\nu x.\varphi_2)$$

Propositions p are in a set Π', variables x are in a set X, and teams I are in $\wp(\Sigma')$.

Given an alternating transition system $\langle \Sigma, Q, \Delta, \Pi, \pi \rangle$ such that $\Pi' \subseteq \Pi$ and $\Sigma = \Sigma'$, with $\mathcal{E} : X \to 2^Q$ a mapping from the variables to the set of states, each formula φ defines a set of states $[\![\varphi]\!]_\mathcal{E}$, computable as follows:

$$\begin{array}{l}
[\![p]\!]_\mathcal{E} = \pi(p) \\
[\![\neg p]\!]_\mathcal{E} = Q \backslash \pi(p) \\
[\![x]\!]_\mathcal{E} = \mathcal{E}(x) \\
[\![\varphi_1 \wedge \varphi_2]\!]_\mathcal{E} = [\![\varphi_1]\!]_\mathcal{E} \cap [\![\varphi_2]\!]_\mathcal{E} \\
[\![\varphi_1 \vee \varphi_2]\!]_\mathcal{E} = [\![\varphi_1]\!]_\mathcal{E} \cup [\![\varphi_2]\!]_\mathcal{E} \\
[\![\langle\!\langle I \rangle\!\rangle \bigcirc \varphi_1]\!]_\mathcal{E} = \mathit{CPre}_I([\![\varphi_1]\!]_\mathcal{E}) \\
[\![[\![I]\!] \bigcirc \varphi_1]\!]_\mathcal{E} = \mathit{UPre}_I([\![\varphi_1]\!]_\mathcal{E}) \\
[\![\mu x.\varphi_1]\!]_\mathcal{E} = \mathrm{lfp}\ \lambda\rho.[\![\varphi_1]\!]_{\mathcal{E}[x \mapsto \rho]} \\
[\![\nu x.\varphi_1]\!]_\mathcal{E} = \mathrm{gfp}\ \lambda\rho.[\![\varphi_1]\!]_{\mathcal{E}[x \mapsto \rho]}
\end{array}$$

If φ is closed, $[\![\varphi]\!]_\mathcal{E}$ does not depend on $\mathcal{E}$, and we will write $[\![\varphi]\!]$ for $[\![\varphi]\!]_\mathcal{E}$. Given an set of initial states $\mathcal{I}$ and a closed formula φ, we will try to approximate $\mathcal{I} \cap [\![\varphi]\!]$, or lfp $\lambda X.([\![\varphi]\!] \cap (\mathcal{I} \cup \mathit{Post}(X)))$, rather than $\mathit{Post}^*(\mathcal{I}) \cap [\![\varphi]\!]$.

5.3 Abstraction of Alternating Transitions Systems

The application of abstract interpretation to alternating transitions systems is already developed in [13], in a model-checking point of view. The definitions are adapted to our notations as follows: the concrete lattice $P^\flat$ is here $\wp(Q)$, so $\top^\flat = Q$, $\sqsubseteq^\flat = \subseteq$, $\bot^\flat = \emptyset$, $\sqcap^\flat = \cap$, $\sqcup^\flat = \cup$. $P^\sharp$ is the abstract lattice, and there is a Galois connection $\wp(Q) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} P^\sharp$. We define now the abstract operators $\pi^\sharp$, $\mathit{UPre}^\sharp_I$ and $\mathit{CPre}^\sharp_I$.

For each p in Π', let $\pi^\sharp(p)$ be an element of $P^\sharp$ such that $\pi(p) \subseteq \gamma(\pi^\sharp(p))$[7], and $\pi^\sharp(\overline{p})$ an element of $P^\sharp$ such that $Q \backslash \pi(p) \subseteq \gamma(\pi^\sharp(\overline{p}))$.

For each subset I of Σ, we define the abstract controllable predecessor relation $\mathit{CPre}^\sharp_I \in P^\sharp \to P^\sharp$ and the abstract uncontrollable predecessor relation $\mathit{UPre}^\sharp_I \in P^\sharp \to P^\sharp$. These relations must satisfy, $\forall \sigma \subseteq Q$:

$$\alpha \circ \mathit{CPre}_I(\sigma) \sqsubseteq^\sharp \mathit{CPre}^\sharp_I \circ \alpha(\sigma)$$
$$\alpha \circ \mathit{UPre}_I(\sigma) \sqsubseteq^\sharp \mathit{UPre}^\sharp_I \circ \alpha(\sigma)$$

These operators are not those used in [13] for the abstract model checking algorithm. The authors use under approximation of concrete relation to obtain a sound abstract model checker. In this paper, we use upper approximation.

5.4 Combining forward and backward abstractions

We need an abstract successor operator for forward analysis. This *abstract successor relation* $\mathit{Post}^\sharp$ must satisfy:

$$\alpha \circ \mathit{Post}(\sigma) \sqsubseteq^\sharp \mathit{Post}^\sharp \circ \alpha(\sigma)$$

[7] If $\alpha(\pi(p))$ is computable, we can take $\pi^\sharp(p) = \alpha(\pi(p))$

Again, we define $Post^{*\sharp} = \lambda X.\mathrm{lfp}\,\lambda Y.(X \sqcup^\sharp Post^\sharp(Y))$. One can easily check that:

$$\alpha \circ Post^*(\sigma) \sqsubseteq^\sharp Post^{*\sharp} \circ \alpha(\sigma)$$

We consider the closed $A\mu$ formulas φ generated by the grammar:

$$\begin{aligned}\varphi ::= {} & p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \langle\!\langle I \rangle\!\rangle \bigcirc \varphi_1 \mid [\![I]\!] \bigcirc \varphi_1 \\ & \mid \sigma x.(\varphi \vee \textstyle\bigvee_{I \in \wp(\Sigma)\setminus\{\emptyset\}} (\varphi_I \wedge \langle\!\langle I \rangle\!\rangle \bigcirc x) \vee \textstyle\bigvee_{I' \in \wp(\Sigma)\setminus\{\emptyset\}} (\varphi_{I'} \wedge [\![I']\!] \bigcirc x))\end{aligned}$$

with $\sigma \in \{\mu, \nu\}$. As before, the last term of the grammar can be rewritten exchanging $\vee$ and $\wedge$ without modifying the expressivity of the logic.

As for the non-game case, we can now define, if φ is a formula, $\Omega_\varphi \in P^\sharp \to P^\sharp$ as follows:

$$\begin{aligned}
\Omega_p(S) &= \pi^\sharp(p) \sqcap^\sharp S \\
\Omega_{\neg p}(S) &= \pi^\sharp(\overline{p}) \sqcap^\sharp S \\
\Omega_{\varphi_1 \wedge \varphi_2}(S) &= \Omega_{\varphi_1}(S) \sqcap^\sharp \Omega_{\varphi_2}(S) \\
\Omega_{\varphi_1 \vee \varphi_2}(S) &= \Omega_{\varphi_1}(S) \sqcup^\sharp \Omega_{\varphi_2}(S) \\
\Omega_{\langle\!\langle I \rangle\!\rangle \bigcirc \varphi}(S) &= CPre^\sharp_I(\Omega_\varphi(Post^\sharp(S))) \\
\Omega_{[\![I]\!] \bigcirc \varphi}(S) &= UPre^\sharp_I(\Omega_\varphi(Post^\sharp(S))) \\
\Omega_{\sigma x.(\varphi \vee \bigvee_I (\varphi_I \wedge \langle\!\langle I \rangle\!\rangle \bigcirc x) \vee \bigvee_{I'} (\varphi_{I'} \wedge [\![I']\!] \bigcirc x))}(S) &= \\
& L^\sharp_{\mathrm{lgfp}}(\, \lambda X.(S \sqcup^\sharp Post^\sharp(X)), \\
& \qquad \lambda Y.(\, \Omega_\varphi(Post^{*\sharp}(S)) \\
& \qquad\qquad \sqcup^\sharp \textstyle\bigsqcup^\sharp_I (\Omega_{\varphi_I}(Post^{*\sharp}(S)) \sqcap^\sharp CPre^\sharp(Y)) \\
& \qquad\qquad \sqcup^\sharp \textstyle\bigsqcup^\sharp_{I'} (\Omega_{\varphi_{I'}}(Post^{*\sharp}(S)) \sqcap^\sharp UPre^\sharp(Y))))
\end{aligned}$$

Theorem 2. *For all formula φ generated by the grammar above, and $\mathcal{I} \subseteq Q$:*

$$\alpha(\mathcal{I} \cap [\![\varphi]\!]) \sqsubseteq^\sharp \alpha(\mathcal{I}) \sqcap^\sharp \Omega_\varphi(\alpha(\mathcal{I}))$$

Proof. The proof is essentially the same of the non-game case.

All we need are the equalities $\mathcal{I} \cap UCPre_I([\![\varphi_1]\!]) \subseteq UCPre_I(Post(\mathcal{I} \cap [\![\varphi_1]\!]))$ with $UCPre = UPre$ or $CPre$, and the equivalence:

$$Y \subseteq b^\flat(X) \Longleftrightarrow Y \subseteq b^\flat(X \cap Post(Y))$$

with $b^\flat = \lambda X.(A \cup \bigcup_I (B_I \cap CPre_I(X)) \cup \bigcup_{I'} (C_{I'} \cap UPre_{I'}(X)))$. These properties are quite easy to check.

6 A simple example

We illustrate the combination with a very short and easy example. We will analyze this small non-deterministic program:

```
(0) { x = 1 }
(1)
    while (n>0) do {
(2)
        if (random in [0,1]=0) then
(3)
            x = x * n;
(4)
        else
(5)
            x = x * (n-1);
(6)
        fi
(7)
        n = n - (input in [0,1]);
(8)
    }
(9)
```

Here, `x`, `n` are integers, `(random in [0,1])` returns a random integer in $[0,1]$, and `(input in [0,1])` returns a integer in $[0,1]$ entered by the user (these commands behave in the same way in the transition relation). Control point (0) is the program entry, we differentiate it from control point (1), which is the while loop entry.

With initial condition `x=1` at control point (0), we will try to prove that the user cannot be sure to have `x=0` at control point (9), that is, the initial condition satisfies $\nu x.((\neg A) \wedge (B \vee \Diamond x) \wedge (C \vee \Box x))$ with A meaning that `x=0` at control point (9), C being the set of states at control point (2), and B being the set of states at other control points.

As we use an upper approximation, we take the negation of the proposition, that is (knowing that $\neg B = C$) : $\mu x.(A \vee (B \wedge \Diamond X) \vee (C \wedge \Box X))$. So we must approximate lfp $\lambda x.([\![A]\!] \cup ([\![B]\!] \cap pre(x)) \cup ([\![C]\!] \cap \widetilde{pre}(x)))$.

We will use *interval analysis* [6], with the improvement of the results of local decreasing iterations [11] for assignments in the backward analysis.

We must abstract $post(X)$, $pre(X)$ and $\widetilde{pre}(X)$. Abstract operators may be described as systems of semantics equations [4, 5]. The program is almost deterministic, and $\widetilde{pre}^\sharp$ is very close to $pre^\sharp$. The differences appear at control points (2) and (7), but we only need to express it at control point (2), with the equation:

$$P_2 = P_3 \sqcap P_5$$

($\sqcap$ being the intersection of abstract environments).

The following table gives the results with a single forward analysis ($F^\sharp(\top^\sharp)$), a single backward analysis ($B^\sharp(\top^\sharp)$), the intersection of both analyses ($F^\sharp(\top^\sharp) \sqcap^\sharp B^\sharp(\top^\sharp)$), and the first iteration of combination ($B^\sharp(F^\sharp(\top^\sharp))$):

Lab. (var.)	$F^\sharp(\top^\sharp)$	$B^\sharp(\top^\sharp)$	$F^\sharp(\top^\sharp) \sqcap^\sharp B^\sharp(\top^\sharp)$	$B^\sharp(F^\sharp(\top^\sharp))$
(0) x:	$[1]$	$[-\infty,+\infty]$	$[1]$	$\emptyset$
n:	$[-\infty,+\infty]$	$[-\infty,+\infty]$	$[-\infty,+\infty]$	$\emptyset$
(1) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[-\infty,+\infty]$	$[-\infty,+\infty]$	$[-\infty,+\infty]$	$[-\infty,+\infty]$
(2) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[1,+\infty]$	$[-\infty,+\infty]$	$[1,+\infty]$	$[1,+\infty]$
(3) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[1,+\infty]$	$[-\infty,+\infty]$	$[1,+\infty]$	$[1,+\infty]$
(4) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[1,+\infty]$	$[-\infty,+\infty]$	$[1,+\infty]$	$[1,+\infty]$
(5) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0,+\infty]$
n:	$[1,+\infty]$	$[-\infty,+\infty]$	$[1,+\infty]$	$[1,+\infty]$
(6) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[1,+\infty]$	$[-\infty,+\infty]$	$[1,+\infty]$	$[1,+\infty]$
(7) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[1,+\infty]$	$[-\infty,+\infty]$	$[1,+\infty]$	$[1,+\infty]$
(8) x:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0]$
n:	$[0,+\infty]$	$[-\infty,+\infty]$	$[0,+\infty]$	$[0,+\infty]$
(9) x:	$[0,+\infty]$	$[0]$	$[0]$	$[0]$
n:	$[-\infty,+\infty]$	$[-\infty,+\infty]$	$[-\infty,+\infty]$	$[-\infty,+\infty]$

The next iteration of the combination will lead to $\emptyset$ everywhere, which is of course the abstract fixpoint $L^\sharp$. So $L^\flat = \emptyset$ (which is not equal to $F^\flat(\top^\flat) \cap B^\flat(\top^\flat)$). As, for this kind of temporal property, $L^\flat \cap \mathcal{I} = \mathcal{I} \cap B^\flat(\top^\flat)$, we obtained the expected result.

7 Conclusion

We have proved that the combination of forward and backward analyses still holds to check *complex temporal properties.* Whereas this combination is useless when dealing with finite domains, and not very useful when abstractions are done by hand (as in the model-checking approach), we expect it will significantly enhance results given by an automatic abstract analyzer of temporal properties.

Using the results of this article will require to have over-approximations of the $\widetilde{pre}$ operator (or predecessor operators of game logic), something which has not been not much studied until now. We also need a method to compute over-approximations of greatest fixpoints since lower narrowing operators give poor results.

Acknowledgements

I wish to thank the anonymous referees, as well as Radhia and Patrick Cousot, Francesco Logozzo for their comments and suggestions.

References

1. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.
2. F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *Proceedings of SIGPLAN '93 Conference on Programming Language Design and Implementation*, pages 46–55, 1993.
3. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking.* MIT press, 1999.
4. P. Cousot. *Méthodes itératives de construction et d'approximation de point fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes.* Thèse ès sciences mathématiques, University of Grenoble, March 1978.
5. P. Cousot. Semantic foundations of program analysis. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.
6. P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106–130. Dunod, Paris, France, 1976.
7. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.
8. P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.
9. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY.
10. P. Cousot and R. Cousot. Temporal abstract interpretation. In *Conference Record of the Twentyseventh Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 12–25, Boston, Mass., 2000. ACM Press, New York, NY.
11. P. Granger. Improving the results of static analyses of programs by local decreasing iterations. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science, 12th conference, New Dehli, India*, volume 652 of *Lecture Notes in Computer Science*, pages 68–79. Springer-Verlag, 1992.
12. T.A. Henzinger, O. Kupferman, and S. Qadeer. From *pre*historic to *post*modern symbolic model checking. In A.J. Hu and M.Y. Vardi, editors, *CAV 98: Computer-aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 195–206. Springer-Verlag, 1998.
13. T.A. Henzinger, R. Majumdar, F.Y.C. Mang, and J.-F. Raskin. Abstract interpretation of game properties. In J. Palsberg, editor, *SAS 00: Static Analysis*, volume 1824 of *Lecture Notes in Computer Science*, pages 220–239. Springer-Verlag, 2000.