Separation — past, present, and future

Ian Hodkinson and Mark Reynolds

1 Introduction

Separation is a remarkable concept, invented by Dov Gabbay in [Gabbay, 1981a], and elaborated in [Gabbay, 1989; Gabbay *et al.*, 1994]. Roughly, a temporal logic has the separation property if its formulas can be equivalently rewritten as a boolean combination of formulas, each of which depends only on the past, present, or future. This seemingly innocent and technical definition has had some far-reaching consequences, and has taken on a life of its own. Surprisingly, separation is closely connected to the important topic of *expressive completeness,* and is one of the main methods for proving expressive completeness. Separation has applications in *executable temporal logic,* and parts of this have been implemented. Separation has found recent uses in simplifying normal form theorems and axiomatising Ockhamist branching time logic, and in analysing the W3C language XPath.

Separation has attracted attention from the time it was introduced. Its simplicity and naturalness have made it appealing to many, and its technical intricacies are still providing employment today.

Reynolds first came across separation through Gabbay in around 1984, and gave a talk on it to Wilfrid Hodges' group at Queen Mary College London, thereby introducing it to Hodkinson who was a Ph.D. student there. Hodkinson and especially Reynolds worked on it with Gabbay during the preparation of the monograph [Gabbay *et al.*, 1994], and it has caught our attention several times since. In this short article, we are happy to return to the topic. We will discuss the original application of separation, some more recent activity on it, and some open problems about it. We thank Carsten Lutz for his ideas and interest, which have improved the 'Future' section.

We would like to take this opportunity to wish Dov a very happy birthday and convey our hopes that his own future will be as prosperous as his past and present.

2 Separation past

Here, we recall the definition of separation itself, and its connection to expressive completeness — the context in which it was first introduced.

2.1 Basic temporal logic

We have to begin with some definitions and notation. They are well known and mostly taken from [Gabbay *et al.*, 1994]. Probably, they can be skipped by those who are familiar with the field and willing to work out the notation at sight. (Temporal logic is the province of philosophers, linguists, and computer scientists, and there seems to be no *standard* notation in the field yet.)

A *flow of time* is a pair $(T, <)$, where T is a non-empty set (the 'time points') and $<$ is an irreflexive transitive relation on T. Examples: $(\mathbb{N}, <)$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$; $(\mathbb{Z}, <)$, where $\mathbb{Z} = \{\ldots -1, 0, 1, 2, \ldots\}$; $(\mathbb{Q}, <)$, where $\mathbb{Q}$ is the set of rational numbers; $(\mathbb{R}, <)$, where $\mathbb{R}$ is the set of real numbers; and various non-linear flows such as trees. Flows of time are (special) Kripke frames. The idea of $<$ is that $t < u$ means that u is a *later* time than t. We will be using *classes* of flows of time, such as the linear flows, dense linear flows, and Dedekind complete linear flows.

We fix a set L of propositional atoms; we write $p, q, r, \ldots$ for atoms. An *assignment* is a map $h : L \to \wp(T)$. For a flow of time $(T, <)$ and an assignment h, the triple $(T, <, h)$ is called a *temporal structure.* The idea of the assignment is that the atom $p \in L$ is *true in* $(T, <, h)$ at the time points in $h(p)$, and *false* at all other times.

We can create *temporal formulas* from the atoms using the boolean operations $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$, $\top$, $\bot$, and *temporal connectives.* The main connectives are F, P, G, H, U, S, T, Y, standing for *at some future (past) time, always in the future (past), Until, Since, Tomorrow,* and *Yesterday,* respectively. The use of the symbols F, P, G, H in this context is due to Prior, and U, S to Kamp; we are not sure about the origin of T, Y. All but U, S are unary (they take a single formula as argument); U, S are binary (they connect two formulas). So, if α, β are temporal formulas, then $F\alpha$, $G\alpha$, $T\alpha$, $U(\alpha, \beta)$, $\neg U(\neg S(\alpha \wedge \beta, \beta \to H\neg\alpha), T(\alpha \vee \beta))$, etc., are also formulas.

The semantics of formulas is defined by induction as follows. Let $\mathcal{M} = (T, <, h)$ be a temporal structure, and $t \in T$ a time point.

1. $\mathcal{M}, t \models p$ iff $t \in h(p)$, for $p \in L$,

2. the booleans are handled as usual,

3. $\mathcal{M}, t \models F\alpha$ iff $\mathcal{M}, u \models \alpha$ for some $u \in T$ with $u > t$,

4. $P\alpha$ is treated similarly using $u < t$ — i.e., the *mirror image* of the preceding clause,
5. $\mathcal{M}, t \models G\alpha$ iff $\mathcal{M}, u \models \alpha$ for all $u > t$,
6. $H\alpha$ — the mirror image clause,
7. $\mathcal{M}, t \models U(\alpha, \beta)$ iff there is $u > t$ with $\mathcal{M}, u \models \alpha$ and $\mathcal{M}, v \models \beta$ for all $v \in T$ with $t < v < u$,
8. $S(\alpha, \beta)$ — mirror image,
9. $\mathcal{M}, t \models T\alpha$ iff there is $u > t$ with $\mathcal{M}, u \models \alpha$ and there is no $v \in T$ with $t < v < u$,[1]
10. $Y\alpha$ — mirror image.

If desired, we can add more connectives with 'first-order-style' definitions as above. On the other hand, sometimes we want to restrict to certain sets $\mathcal{T}$ of connectives. For simplicity, we will generally call such a $\mathcal{T}$ a *temporal logic*. For example, we refer to 'the temporal logic US', by which we mean the formulas written with the booleans and the connectives U and S.

In this article we are only concerned with syntax and semantics, not with any kind of reasoning. But it will already be obvious that there are some connections between formulas. $H\alpha$ and $\neg P \neg \alpha$ 'mean the same'; so do $F\alpha$ and $U(\alpha, \top)$, and $T\alpha$ and $U(\alpha, \bot)$. On the other hand, $YU(\alpha, \beta)$ will mean the same as $\alpha \vee (\beta \wedge U(\alpha, \beta))$ in flows of time like $(\mathbb{Z}, <)$, but not in dense flows like $(\mathbb{Q}, <)$. We formalise this 'relativity' as follows.

DEFINITION 1. Given a class $\mathcal{C}$ of flows of time, we say that temporal formulas α, β are *equivalent over* $\mathcal{C}$ if $(T, <, h), t \models \alpha \leftrightarrow \beta$ for all $(T, <) \in \mathcal{C}$, all assignments $h : L \to \wp(T)$ and all $t \in T$.

We write 'equivalent over linear time' to abbreviate 'equivalent over the class of all linear flows of time', etc.

2.2 Standard translation and expressive completeness

As in modal logic, temporal formulas can be translated to first-order ones.

DEFINITION 2. Let L' be the first-order signature consisting of $<$, and unary relation symbols $P, Q, \ldots$ twinned with the atoms $p, q, \ldots \in L$, respectively.

[1] Note that in dense flows of time like $(\mathbb{Q}, <)$, $T\alpha$ is always false; but it is not 'meaningless'.

Each temporal formula α has a 'standard translation', which is a first-order L'-formula $\alpha^x(x)$ with free variable x (for any first-order variable x), as follows.

1. Each atom p translates to $p^x(x) = P(x)$.
2. The translation $-^x$ commutes with the booleans.
3. $(F\alpha)^x = \exists y(y > x \wedge \alpha^y)$.
4. $(G\alpha)^x = \forall y(y > x \to \alpha^y)$.
5. $U(\alpha, \beta)^x = \exists y(y > x \wedge \alpha^y \wedge \forall z(x < z < y \to \beta^z))$.
6. The others are similar.

It is clear that we can view a temporal structure $(T, <, h)$ as an L'-structure by interpreting $P, Q, \ldots$ as $h(p)$, $h(q), \ldots$, respectively, and that by so doing, we have

$$(T, <, h), t \models \alpha \text{ iff } (T, <, h) \models \alpha^x[t]$$

for all $t \in T$ and all formulas α. *The translation $\alpha \mapsto \alpha^x$ is meaning-preserving.* So it makes good sense to extend definition 1:

DEFINITION 3. A temporal formula α is said to be *equivalent* to an L'-formula $\varphi(x)$ over a class $\mathcal{C}$ of flows of time if for all $(T, <) \in \mathcal{C}$ and all $h : L \to \wp(T)$, we have $(T, <, h) \models \forall x(\alpha^x \leftrightarrow \varphi(x))$.

Not all first-order formulas $\varphi(x)$ will have the form α^x for some temporal formula α. Surprisingly however, sometimes every $\varphi(x)$ is *equivalent* to some α.

DEFINITION 4. We say that a temporal logic $\mathcal{T}$ is *expressively complete over a class $\mathcal{C}$ of flows of time* if for every L'-formula $\varphi(x)$, there is a $\mathcal{T}$-formula α that is equivalent to φ over $\mathcal{C}$.

Kamp proved in the very important [Kamp, 1968] that U and S are expressively complete over Dedekind-complete linear time. This first expressive completeness theorem led to a canon of results, continuing today, and it brings our story finally to separation and Gabbay's contribution.

2.3 Separation

We now explain formally what separation is. Some jargon will be handy for this. Given a flow of time $(T, <)$, a time $t \in T$, and assignments $g, h : L \to \wp(T)$, we say that

- *g, h agree on t* if for all $p \in L$, we have $t \in g(p)$ iff $t \in h(p)$.

- *g and h agree on the past of t* if for all $u \in T$ with $u < t$, and all $p \in L$, we have $u \in g(p)$ iff $u \in h(p)$.
- *'g and h agree on the future of t'* is defined by the mirror image of the preceding clause.

DEFINITION 5. [Gabbay, 1981a] Let $\mathcal{C}$ be a class of flows of time. A temporal formula α is said to be

- *pure past over* $\mathcal{C}$, if for any $(T, <) \in \mathcal{C}$ and any $t \in T$, whenever $g, h : L \to \wp(T)$ are assignments that agree on the past of t, we have $(T, <, g), t \models \alpha$ iff $(T, <, h), t \models \alpha$;
- *pure present over* $\mathcal{C}$, if for any $(T, <) \in \mathcal{C}$, $t \in T$, and assignments $g, h : L \to \wp(T)$ that agree on t, we have $(T, <, g), t \models \alpha$ iff $(T, <, h), t \models \alpha$;
- *pure future over* $\mathcal{C}$, if for any $(T, <) \in \mathcal{C}$, any $t \in T$, and any assignments $g, h : L \to \wp(T)$ agreeing on the future of t, we have $(T, <, g), t \models \alpha$ iff $(T, <, h), t \models \alpha$;
- *pure over* $\mathcal{C}$, if it is pure past, pure present, or pure future over $\mathcal{C}$,
- *separated over* $\mathcal{C}$, if it is a boolean combination of formulas that are pure over $\mathcal{C}$.

$\mathcal{T}$ is said to have the *separation property over* $\mathcal{C}$ if every $\mathcal{T}$-formula is equivalent over $\mathcal{C}$ to a formula that is separated over $\mathcal{C}$.

So a formula is pure past if its truth value at any time depends only on the values of its atoms in the past; and similarly for pure present and pure future. Hq and $S(p,\ q \to S(q, r))$ are pure past formulas. The formula $F(q \wedge Hr)$ is not pure, but it is equivalent over linear time to the separated formula $Hr \wedge r \wedge U(q, r)$.

2.4 Gabbay's theorem

We are now ready to state the important result of Gabbay that relates separation to expressive completeness. It was stated in [Gabbay, 1981a, theorem 11] and proved in [Gabbay, 1989] and [Gabbay *et al.*, 1994, §9.3]. The theorem is surprising because it connects two seemingly different conditions.[2]

THEOREM 6. *Let $\mathcal{C}$ be a class of linear flows of time, and $\mathcal{T}$ a temporal logic able to express F and P over $\mathcal{C}$. Then $\mathcal{T}$ is expressively complete over $\mathcal{C}$ iff it has the separation property over $\mathcal{C}$.*

[2] The legend goes that when Kamp heard this theorem in a seminar, he went out and bought Dov a cake.

Proof '⇒' is proved by showing that any first-order L'-formula can be separated over linear time. Here, we sketch a proof running along standard model-theoretic lines.[3] It is entirely 'classical', involving no temporal logic.

First observe that we can define equivalence, purity, and separatedness for *L'-formulas* $\varphi(x)$ semantically (over linear time). The definitions are analogous to those for temporal formulas (definitions 1 and 5).

Fix an L'-formula $\varphi(x)$ with free variable x, and let $L_\varphi \subseteq L'$ be the finite relational signature consisting of $<$ and the unary relation symbols that occur in φ. An L_φ-structure has the form $(T, <, h)$, where h is a map that assigns each unary relation symbol occurring in φ to a subset of T.

We want to show that there is a separated L_φ-formula that is equivalent to $\varphi(x)$ over linear time (that is, over all L_φ-structures $(T, <, h)$ where $(T, <)$ is a linear flow of time). We do this using compactness and games.

Let λ be a sentence saying that $<$ is an irreflexive linear order. Let $\Sigma(x)$ be the set of all separated L_φ-formulas $\sigma(x)$ implied by φ over linear time: formally, those such that $\lambda \vdash \forall x(\varphi(x) \to \sigma(x))$.

We first show that $\Sigma \models \varphi$ over linear time. So suppose that $(T, <, h) \models \Sigma[t]$, where $(T, <)$ is a linear flow of time, and $t \in T$. We show that $(T, <, h) \models \varphi[t]$.

Let $\Theta(x)$ be the set consisting of

- all pure L_φ-formulas $\pi(x)$ such that $(T, <, h) \models \pi[t]$,
- $\lambda \wedge \varphi(x)$.

We claim that Θ is consistent. If it were not, then by first-order compactness, there would be pure $\pi_1(x), \ldots, \pi_n(x) \in \Theta$ with $(T, <, h) \models \pi_i[t]$ for each i, such that $\lambda \vdash \forall x(\varphi(x) \to \neg \bigwedge_{i \leq n} \pi_i(x))$. But then, $\neg \bigwedge_{i \leq n} \pi_i(x) \in \Sigma$ and so $(T, <, h) \models \neg \bigwedge_{i \leq n} \pi_i[t]$. This is a contradiction, and establishes the consistency of Θ.

So there is a model $(T', <', h') \models \Theta[t']$, where $(T', <')$ is a linear flow of time and $t' \in T'$. In short, $(T, <, h, t)$ and $(T', <', h', t')$ are linear, agree on all pure L_φ-formulas, and φ is true in the latter.

But now, if $\pi(x)$ is a pure past L_φ-formula then $(T, <, h) \models \pi[t]$ iff $(T', <', h') \models \pi[t']$. Since we can relativise (the quantifiers in) any L_φ-sentence to the points in the past of t, so obtaining a pure past formula, this means that the substructures of $(T, <, h)$ and $(T', <', h')$ consisting of all time points in the past of t, t', respectively, are elementarily equivalent. The same holds for pure present and pure future formulas, so the substructures based on t, t' are elementarily equivalent, and the substructures based on points in the future of t, t' are elementarily equivalent too. Now an Ehrenfeucht–Fraïssé game

[3] For a more effective proof, see [Gabbay *et al.*, 1994, 9.3.2].

argument (which is legal because L_φ is finite), or the Feferman–Vaught theorem (cf. [Hodges, 1993, A.6.2]), will easily show that $(T, <, t, h)$ and $(T', <', t', h')$ are themselves elementarily equivalent. Since φ is true in the second structure, we have $(T, <, h) \models \varphi[t]$ as required.

We know now that $\Sigma \models \varphi$ over linear time. By compactness, and as linearity is first-order definable, there is a conjunction $\sigma(x)$ of separated formulas in Σ that implies φ over linear time. Hence, $\varphi(x)$ and $\sigma(x)$ are equivalent over linear time. Of course, σ is separated.

Having got through this first stage, it is now easy to show that $\mathcal{T}$ has the separation property over $\mathcal{C}$. Let α be a $\mathcal{T}$-formula. By the above, its standard translation α^x is equivalent over linear time, and so over $\mathcal{C}$, to a separated formula — a boolean combination of pure formulas. But $\mathcal{T}$ is expressively complete over $\mathcal{C}$, so each of these pure formulas is equivalent over $\mathcal{C}$ to a $\mathcal{T}$-formula, which is obviously pure as well. The corresponding boolean combination of these $\mathcal{T}$-formulas is separated and equivalent to α over $\mathcal{C}$. This completes the proof.

For '$\Leftarrow$', assume that $\mathcal{T}$ has the separation property over $\mathcal{C}$. We need to show that any first-order L'-formula $\varphi(x, P_1, \ldots, P_n)$, with one free variable x and unary relation symbols $P_1, \ldots, P_n$, is equivalent over $\mathcal{C}$ to some $\mathcal{T}$-formula. We go by induction on the quantifier depth k of φ.

If k is 0, then φ is a boolean combination of formulas of the form $P_i(x)$, $x = x$, and $x < x$, so the result is clear. Assume the result for k. It suffices to express $\exists y \varphi(x, y, P_1, \ldots, P_n)$ as a $\mathcal{T}$-formula, where φ has quantifier depth k. We can suppose that x and y do not occur bound in φ.

First, we want to remove x from φ somehow, to obtain a formula $\varphi'(y)$ to which we can apply the inductive hypothesis. How can we do it? Let us start by thinking about the atomic subformulas of φ involving x. They are of the form $P_i(x)$, $x = x$, $x < x$, $z = x$, $x = z$, $z < x$, and $x < z$, where z is some other variable than x. We can replace $x = z$ by $z = x$ and leave it for later. We replace $x = x$ by $\top$, and $x < x$ by $\bot$. This leaves $z < x$, $z = x$, $x < z$, and the $P_i(x)$.

Next, we root out the $P_i(x)$. The idea here is based on a simple observation about propositional logic, exemplified in the statement that an arbitrary propositional formula like $\neg p \wedge q$ is equivalent to

$$\begin{array}{rcl} (p & \rightarrow & \neg\top \wedge q) \\ \wedge\ (\neg p & \rightarrow & \neg\bot \wedge q). \end{array}$$

The right-hand sides here do not involve p.

Proceeding in this way, for each $S \subseteq \{1, 2, \ldots, n\}$, let φ^S be the result of replacing each atomic subformula $P_i(x)$ of φ by $\top$ if $i \in S$, and by $\bot$ if $i \notin S$. Then φ^S still has the same quantifier depth, k, but has no occurrences of

any $P_i(x)$. (It may involve $P_i(z)$ for other variables z.) We see that $\exists y\varphi$ is equivalent to

$$\bigwedge_{S\subseteq\{1,\dots,n\}} \Big(\bigwedge_{i\in S} P_i(x) \wedge \bigwedge_{j\notin S} \neg P_j(x) \to \exists y \varphi^S(x, y, P_1, \dots, P_n) \Big).$$

The formulas $P_i(x)$ are equivalent over $\mathcal{C}$ to p_i, of course. So it is enough if we can express the $\exists y\varphi^S(x, y, P_1, \dots, P_n)$ as $\mathcal{T}$-formulas.

Hence, we can assume that the original formula φ is such a formula. That is, we suppose that x does not occur in atomic subformulas of φ of the form $P_i(x)$, but only in ones of the form $z = x$, $z < x$, and $x < z$.

It remains to get rid of these too. To do it, we temporarily introduce new atoms $n_=$, $n_<$, $n_>$, with corresponding unary relation symbols $N_=$, etc. We replace each atomic subformula $z = x$ in φ (where z is any variable) by $N_=(z)$. Similarly, replace $x < z$ by $N_>(z)$, and $z < x$ by $N_<(z)$. This yields a formula $\varphi'(y, P_1, \dots, P_n, N_<, N_=, N_>)$ of quantifier depth k and with no occurrences of x at all. For a while, we will restrict ourselves to structures interpreting $N_=$ as $\{x\}$, $N_<$ as $\{t : t < x\}$, and $N_>$ as $\{t : t > x\}$. Then each $N_*(z)$ is equivalent to the formula it replaced, so $\exists y\varphi(x, y, P_1, \dots, P_n)$ will be equivalent to $\exists y\varphi'(y, P_1, \dots, P_n, N_<, N_=, N_>)$.

Inductively, $\varphi'(y, P_1, \dots, P_n, N_<, N_=, N_>)$ is equivalent over $\mathcal{C}$ to some $\mathcal{T}$-formula $\alpha(p_1, \dots, p_n, n_<, n_=, n_>)$. The flows in $\mathcal{C}$ are linear, so $\exists y\varphi'$ is equivalent over $\mathcal{C}$ to $\beta = \alpha \vee F\alpha \vee P\alpha$ (here we use the hypothesis that F and P are $\mathcal{T}$-expressible). Thus, *under our restriction,* the original formula $\exists y\varphi(x, y, P_1, \dots, P_n)$ is equivalent over $\mathcal{C}$ to $\beta(p_1, \dots, p_n, n_<, n_=, n_>)$.

Finally, we come to the key step. We remove the temporary n-atoms from β. We do this using the separation property! We separate β, obtaining a boolean combination $\gamma(p_1, \dots, p_n, n_<, n_=, n_>)$ of *pure* $\mathcal{T}$-formulas. Consider, say, a *pure past* formula $\delta(p_1, \dots, p_n, n_<, n_=, n_>)$ from this boolean combination. As δ is pure past, its truth value at x only depends on the values of its atoms at points $t < x$. It is independent of their values at points $t \geq x$. But under our restriction, the values of the n-atoms at points $< x$ is entirely predictable: $n_<$ is true, and the others are false. Accordingly, let us replace $n_<$ in δ by $\top$, and replace $n_=$ and $n_>$ by $\bot$. We obtain $\delta^*(p_1, \dots, p_n) = \delta(p_1, \dots, p_n, \top, \bot, \bot)$. Then, always under our restriction, the truth values of δ^* and δ at x are the same.

We adjust each pure formula δ in γ in a similar way. If δ is pure present, $n_=$ is replaced by $\top$, and the others by $\bot$. For pure future δ, $n_>$ is replaced by $\top$ instead. The result is a boolean combination $\gamma^*(p_1, \dots, p_n)$, which, subject to our restriction, is equivalent to $\exists y\varphi(x, y, P_1, \dots, P_n)$.

But the restriction concerned atoms that do not appear in γ^*. So it has no force. Without any restriction on assignments to atoms, γ^* is equivalent

over $\mathcal{C}$ to $\exists y\varphi(x, y, P_1, \ldots, P_n)$. This completes the induction and the proof. ❑

Given an algorithm to separate a formula over $\mathcal{C}$, the proof above can be elaborated into an algorithm that will translate any given first-order formula $\varphi(x, P_1, \ldots, P_n)$ to a $\mathcal{C}$-equivalent $\mathcal{T}$-formula. However, it appears that the complexity of this algorithm is high. Over $(\mathbb{N}, <)$, the problem is known to have non-elementary complexity: that is, there is no translation algorithm that runs in exponential time, double exponential time, triple exponential time, etc. A stronger result was proved in [Etessami and Wilke, 2000]: that there is no elementary bound on the *length* of a US-formula equivalent to a first-order one. See section 4 for some related problems on the separation process.

2.5 Separation and expressive completeness

Not all temporal logics have the separation property. For example, FP over linear time does not: the formula $F(q \wedge Hr)$ cannot be separated using only F and P. We now discuss some temporal logics that do have the separation property.

THEOREM 7 (Gabbay; see [Gabbay, 1989], [Gabbay *et al.*, 1994, 10.2.9]). *Until and Since have the separation property over* $(\mathbb{N}, <)$ *(i.e., over the class* $\{(\mathbb{N}, <)\}$*).*

Hence, Until and Since are expressively complete over $(\mathbb{N}, <)$. The proof involves an induction showing that each formula can be separated.

THEOREM 8 (Gabbay–Reynolds; see [Gabbay *et al.*, 1994, §§10.3–11]).

1. *Until and Since have the separation property over* $(\mathbb{R}, <)$.
2. *Until, Since, and the Stavi connectives have the separation property over the class of all linear flows of time.*

Hence, U and S are expressively complete over $(\mathbb{R}, <)$, and U, S, and the Stavi connectives are expressively complete over all linear flows. The *Stavi connectives* were mentioned (without definition) in [Gabbay *et al.*, 1980] and are versions of U, S oriented towards Dedekind cuts in the flow of time; we refer to [Gabbay, 1981a; Gabbay *et al.*, 1994] for details.

The proof of theorem 8 is broadly similar to that of theorem 7, but more complicated. Both proofs are effective. The spirit of the proofs suggests a 'heuristic' to tell whether a given set of connectives is expressively complete: as Gabbay said, *'try to separate and see where you get stuck!'* This may inspire the addition of extra connectives that are expressively complete. For example, trying to separate $F(q \wedge Hr)$ over linear time shows the need for

Until. We already said that this formula cannot be separated using only F and P. But a look at what the formula says suggests the addition of $U(q, r)$ to express what it says about the future. Then $F(q \wedge Hr)$ is equivalent to $Hr \wedge r \wedge U(q, r)$, which is separated over linear time.

The concept of separation is not confined to linear time; generalisations to non-linear flows have been known since the 1980s. The idea is to break up the flow of time into suitable 'regions' on which the values of the atoms $n_<, n_=, n_>$ are constant, and that allow the Ehrenfeucht–Fraïssé game argument of theorem 6 to go through. Various separation and expressive completeness results for trees were proved in [Amir, 1982a; 1982b; 1985; Amir and Gabbay, 1987; Immerman and Kozen, 1987; Schlingloff, 1992].

2.6 Syntactic separation

The proof of theorem 7 actually shows that over $(\mathbb{N}, <)$, every US-formula α is equivalent to a boolean combination β of U-formulas and S-formulas. We call such a β *syntactically separated.* This is formally stronger than a simple 'semantic' separation result, and it can be very useful to have. But on the theoretical level, it is only a marginal improvement. This is for two reasons.

First, while recognising a *syntactically* separated formula is trivial, it is not so hard to tell whether a formula is *semantically* pure or separated. The complexity is often the same as for the validity problem, and temporal logic is frequently advertised in terms of the relatively low complexity of its validity problem.

LEMMA 9. *The problems of deciding whether, over* $(\mathbb{N}, <)$*, a given* US*-formula is (a) pure past, (b) pure present, (c) pure future, (d) separated, are* PSPACE*-complete.*

Proof The proof relies on the PSPACE-completeness of the problem of validity of US-formulas over $(\mathbb{N}, <)$ [Sistla and Clarke, 1985].

A US-formula $\alpha(p_1, \ldots, p_n)$ is pure future over $(\mathbb{N}, <)$ iff the formula

$$\Big(G \bigwedge_{i=1}^{n} (p_i \leftrightarrow q_i)\Big) \rightarrow \Big(\alpha(p_1, \ldots, p_n) \leftrightarrow \alpha(q_1, \ldots, q_n)\Big)$$

is valid in $(\mathbb{N}, <)$. This formula is constructible from α in polynomial time. By checking its validity, we decide pure future-ness of α in PSPACE. The other two cases are dealt with similarly. This algorithm is due to Lutz (personal communication, 2005).

Whether α is separated over $(\mathbb{N}, <)$ can then be checked by searching for a boolean decomposition of α into pure formulas; the search can be conducted in PSPACE.

The proof of PSPACE-hardness uses the rather counter-intuitive fact that for any US-formula α and any atom q not occurring in α, the following five conditions are equivalent (all taken over $(\mathbb{N}, <)$): (i) α is valid, (ii) $\alpha \vee GHq$ is pure past, (iiv) $\alpha \vee GHq$ is pure present, (iv) $\alpha \vee GHq$ is pure future, (v) $\alpha \vee GHq$ is separated. This is easily checked. If α is valid, so is $\alpha \vee GHq$, so it is trivially pure and separated. If α is not valid, we can choose $h : L \to \wp(\mathbb{N})$ and $t \in \mathbb{N}$ with $(\mathbb{N}, <, h), t \models \neg\alpha$. Then the value of $\alpha \vee GHq$ at t depends on whether $h(q) = \mathbb{N}$ or not. As this can be varied without changing the value of q in the past of t, we see that $\alpha \vee GHq$ is not pure past. Similarly, it is not pure present or pure future. Nor is $\alpha \vee GHq$ separated, since GHq is not pure. Thus, we have reduced validity in polynomial time to being pure or separated, showing PSPACE-hardness of the latter. ❑

The second reason is that whilst a syntactic separation result is available over $(\mathbb{N}, <)$, it is not possible to obtain one over general flows of time. Over $(\mathbb{R}, <)$ for example, $P\neg U(\top, q)$ is pure past, but is not equivalent to any S-formula. However, lemma 9 generalises easily to $(\mathbb{R}, <)$ using the PSPACE-completeness of validity of US-formulas over $(\mathbb{R}, <)$ proved in [Reynolds, 1999].

In short, we have to live with semantic separation in many cases, and the algorithmic costs of doing so are not too bad.

3 Separation: present

In this section we consider recent and current work involving separation. We also mention briefly some earlier uses of separation which have stood the test of time particularly well: these are cases where a proof or technique which relies on separation still attracts some current research activity.

3.1 Expressive Completeness

As outlined above, separation can demonstrate expressive completeness of temporal languages. It gives a theoretical justification for using a particular language. Separation, along with the construction in the proof of theorem 6 stands as one of the main ways of translating from first-order logic into temporal logic in such important cases as natural numbers and real numbers time.

Other ways of approaching such a task do exist but seem to be even less well developed towards actual implementation. These include the direct syntactical arguments in Kamp [Kamp, 1968], the outline by Stavi in [Gabbay *et al.*, 1980] and the game-theoretic version of this in chapter 12 of [Gabbay *et al.*, 1994].

Of course, this translation is not done as part of any practical reasoning application as it seems to be too expensive.

More recently, Marx noticed a connection with temporal logic and used an original but also quite traditional separation approach to analyse the expressiveness of XPath, the W3C-standard node addressing language for XML documents [Marx, 2004]. He was able to add some natural extra relations to XPath to achieve an expressive completeness result.

3.2 Removing past-time operators

Separation is also called upon to justify *not* using past-time operators in temporal languages which specify systems. It has been well argued [Lichtenstein *et al.*, 1985] that specifications using past-time operators are more natural. It is also clear that it is often the case that including the past-time operators adds no complexity to reasoning tasks [Sistla and Clarke, 1985]. However, across the wide ranging uses of temporal languages we find many examples where past-time operators are neglected. These include reasoning about branching time where the future-only CTL and CTL* are long established. Reasoning about evolving knowledge using temporal-epistemic combinations is another example: the basic languages are set out in [Fagin *et al.*, 1995] and past-time operators are only mentioned briefly to note that they might complicate the exposition.

The justification for not having past-time operators is often not explicit. We would agree with [Lichtenstein *et al.*, 1985] in supposing that researchers are relying on the claim in [Gabbay *et al.*, 1980] that the past-time operators add nothing to the expressivity of the language with Until over the natural numbers time.

The claim (Theorem 2.1 in [Gabbay *et al.*, 1980]), which follows easily from theorem 7, is just that for every formula $\phi(x)$ of the first-order language of order (L' from definition 2) there is a temporal formula from the language with just U which is equivalent to ϕ over $(\mathbb{N}, <)$ at time 0. It follows that

LEMMA 10. *For any US-formula α there is a U-formula β that is equivalent to α at time* 0. *That is, for any $h : L \to \wp(\mathbb{N})$, we have $(\mathbb{N}, <, h), 0 \models \alpha \leftrightarrow \beta$.*

To see this, just syntactically separate α, and replace all maximal subformulas $S(X, Y)$ of the result by **false**. We will call such an equivalence at time zero, *initial equivalence* between formulas.

We should note that, in many computer science oriented publications since [Gabbay *et al.*, 1980], the temporal languages considered use what are called the non-strict versions of U and S. To distinguish these from the strict versions which we introduced in section 2 we write them in an infix manner and as $U_=$ and $S_=$. For example, $M, t \models pU_=q$ iff there is $u \geq t$ with $M, u \models q$ and for all $v \in T$, if $t \leq v < u$ then $M, v \models p$. The language with $U_=$ and T is known as LTL, and if $S_=$ and Y are included then we

might call this LTL+PAST (but the notation is not everywhere consistent).

It is clear that on natural number time structures, $\mathcal{T} = (\mathbb{N}, <, h)$, strict until can be defined by $U(\beta, \alpha) \equiv T(\alpha U_= \beta)$ and, equally, the LTL operators can be defined from U: $\alpha U_= \beta \equiv \beta \vee (\alpha \wedge U(\beta, \alpha))$ and $T\alpha \equiv U(\alpha, \mathbf{false})$. Similarly for strict since.

The claim from [Gabbay *et al.*, 1980] thus tells us that the past-time operators $S_=$ and Y do not add anything to the expressivity of the language with $U_=$ and T, provided we are just interested in the satisfiability of formulas at time zero. For specifying the behaviour of a system, of course, it is natural to just state its desired behaviour as a formula to hold at time zero.

It is not clear that there are any other ways, apart from separation, to remove past-time operators from US-formulas over the natural numbers to make an initially equivalent U-formula.

3.3 Executable temporal logic

Separation stands as one of the foundations on which the intuitively appealing "declarative past implies imperative future" idea of [Gabbay, 1989] is used to allow arbitrary temporal (declarative) specifications to be executed as (imperative) programs. This idea, and executable temporal logics more generally, are used for rapid proto-typing in fields from system modelling [Finger *et al.*, 1993] to multimedia [Bowman *et al.*, 2003] and AI [Jonker and Wijngaards, 2003].

Considering the particular Metatem approach [Barringer *et al.*, 1996] to executable temporal logic, we see that separation gives a theoretical justification for only dealing with specifications given in a "past implies future" normal form. Alternatively, it gives a process for converting an arbitrary specification into this normal form. A Metatem program is of the form

$$G(\bigwedge_{i=1}^{n}(\xi_i \rightarrow \psi_i))$$

where the ξ_i are pure past and the ψ_i are boolean combinations of pure present and pure future formulas. Note that there are further syntactic restrictions. The idea is that on the basis of the declarative truth of the past time ξ_i the program should go on to "do" ψ_i.

The process given in chapter 4 in [Barringer *et al.*, 1996] shows how an arbitrary formula of the propositional language with $U_= S_= YT$ can be converted into an equally satisfiable formula in this normal form. This process starts by assuming that the formula will be separated: it is not easy to see any alternative than to use the separation process described in [Gabbay *et al.*, 1994].

3.4 The safety-liveness normal form

A more recent application for the separation process is in achieving another normal form for temporal formulas. This is the safety-liveness form of [Lichtenstein *et al.*, 1985].

One of the many interesting observations made in [Lichtenstein *et al.*, 1985] is that by using some transformations based on separation, automata and regular expressions one can find, for any temporal formula (possibly involving past operators) an equivalent formula in which the only future operators are unnested and of the form "infinitely often" or "eventually always". This form is used to show that any temporal formula is equivalent to a positive boolean combination of what [Lichtenstein *et al.*, 1985] define as safety and simple liveness properties— safety properties being those that must continue to hold forever, and liveness properties being those which must eventually happen. We thus call this the safety-liveness form.

From [Lichtenstein *et al.*, 1985] we have:

LEMMA 11. *Every formula of LTL+Past is initially equivalent to a formula of the form:*

$$\bigvee_{i=1}^{n} (GF\alpha_i) \wedge (FG\beta_i)$$

where each α_i and β_i are formulas without $U_=$ or X.

The proof in [Lichtenstein *et al.*, 1985] first uses separation to find a LTL equivalent. It then proceeds via standard translations of a LTL formula into an equivalent Büchi automaton (i.e. an automaton that accepts exactly the structures which are models of the formula). The standard translations use a result from [McNaughton, 1966] which allows us to find a deterministic equivalent to any Büchi automaton and so, via a recursive construction, allows us to find a negation of a given automaton.

The automaton will be what is known as counter-free and this allows us (via results from [McNaughton and Papert, 1971]) to find star-free regular expressions which describe various possible limiting behaviours in terms of finite prefixes of the structure. Finally, these are converted into equivalent past-time formulas via a reverse translation.

A newer proof of lemma 11 is given in [Reynolds, 2000]. It also uses separation but relies on a less varied range of other powerful techniques.

We use the stronger separation result appropriate for any Dedekind complete linear order, i.e. $(T, <)$ such that every non-empty subset of T with an upper bound has a least upper bound. This is used to establish theorem 8 above. The result (Theorem 10.3.20, [Gabbay *et al.*, 1994]) involves semantic separation and concepts of purity. Here we just need to extract some ideas from the proof.

We introduce new operators
$K^+(\alpha) = \neg U(\mathbf{true}, \neg\alpha)$,
$K^-(\alpha) = \neg S(\mathbf{true}, \neg\alpha)$,
$\Gamma^+(\alpha) = \neg K^+(\neg\alpha) \wedge K^-(\neg\alpha)$, and
$\Gamma^-(\alpha) = \neg K^-(\neg\alpha) \wedge K^+(\neg\alpha)$
and then add these to the US language to define an extended language, which we call ESTL. $K^+(\alpha)$ holds at a point when α holds arbitrarily soon afterwards and the other connectives have similarly intuitive readings.

In the proof of the theorem, some dozens of ESTL equivalences, or *acceptable rewrites*, are presented. The general idea is that formulas with past operators within the scope of future operators (or vice versa) are rewritten with one less level of such nesting. A recursive procedure is given for eventually eliminating all such nesting via a series of these rewrites. See [Gabbay *et al.*, 1994] for details.

In ESTL, we say that a formula is *syntactically separated* iff it is a boolean combination of formulas that are either atoms or of the form $U(\alpha, \beta)$ and not containing S, $S(\alpha, \beta)$ and not containing U, $K^+(\alpha)$ and not containing S, or $K^-(\alpha)$ and not containing U. Such subformulas are semantically pure (as in definition 5) and we will see that over a certain flow of time, which is nearly everywhere discrete, they have a particularly simple form.

From the proof of Theorem 10.3.20 in [Gabbay *et al.*, 1994] we have the following:

LEMMA 12. *Over Dedekind complete time, each formula in ESTL can be acceptably rewritten as an equivalent formula which is syntactically separated.*

It must be pointed out that despite only involving straightforward syntactic rewrites, the general separation procedure may be rather computationally complex. In fact, the time complexity of the procedure (and hence the size blow-up in formulas) is probably nonelementary.

The new proof from [Reynolds, 2000] of lemma 11 based solely on separation is as follows.

Proof The idea is to add a point ∞ after all the natural numbers to get a new Dedekind complete linear order $\mathbb{N}^\infty$. We will work in the temporal logic with US over propositional structures with $\mathbb{N}^\infty$ as the flow of time.

Our first step is to relativize the US version of our formula ϕ to places where $\kappa = U(\mathbf{true}, \mathbf{true})$ holds (via recursive use of such transformations of $U(\alpha, \beta)$ to $U(\kappa \wedge \alpha, \kappa \rightarrow \beta)$). Say that the result is ϕ^+. It is clear that ϕ is true at 0 in a natural number structure iff ϕ^+ is true at 0 in any $\mathbb{N}^\infty$ extension.

To say that ϕ^+ holds at time 0 is clearly equivalent to saying that

$\beta^+ = S(\neg S(\mathbf{true}, \mathbf{true}) \wedge \phi^+, \mathbf{true})$ holds at time ∞. By using the separation technique Lemma 12, we can find, (effectively via the straightforward syntactic transformation via the set of rewrite rules), an equivalent formula γ^+, which is a boolean combination of syntactically pure ESTL formulas. Obviously we can dispense with the present and future parts of this combination (eg, assume that all atoms are false at time ∞ and any formula $U(\alpha, \beta)$ or $K^+(\alpha)$ can be replaced by falsity).

The next step is to get rid of $K^\pm$ and $\Gamma^\pm$. Note that we can rewrite $K^-(\alpha)$ as $\neg S(\mathbf{true}, \neg\alpha)$ so we need only consider maximal subformulas of γ^+ of the form $S(\alpha, \beta)$ in which α and β contain no U. The fact that $\mathbb{N}^\infty$ is discrete at all points $< \infty$ at which we evaluate such α and β (and their subformulas) allows us to equivalently rewrite each $K^+(\psi)$ and $\Gamma^-(\psi)$ subformula as **false** and each $\Gamma^+(\psi)$ formula as $\neg S(\mathbf{true}, \neg\psi)$. An induction on the construction of each α or β is actually needed here to guarantee that we do only need to evaluate formulas at points $< \infty$ but this is straightforward. Thus we may assume that γ^+ is a boolean combination of US formulas of the form $S(\alpha, \beta)$ containing no U.

So consider a pure past boolean component of γ^+. It will be in the form $\delta = S(\eta, \theta)$ with η and θ being formulas without U. To say that δ holds in the extended structure at time ∞ is just to say that at time 0 in the original structure, $\theta \wedge S(\eta, \theta)$ eventually always holds. To say that δ does not hold in the extended structure at time ∞ is just to say that at time 0 in the original structure, $\neg(\theta \wedge S(\eta, \theta))$ holds infinitely often. By rewriting γ^+ in disjunctive normal form we can extract the required α_i and β_i from conjunctions of such $\theta \wedge S(\eta, \theta)$ and their negations: use the equivalences $FG\alpha \wedge FG\beta \leftrightarrow FG(\alpha \wedge \beta)$ and $GF\alpha \vee GF\beta \leftrightarrow GF(\alpha \vee \beta)$ to collect the conjunctions together. ❑

Obviously the new proof calls on a more limited variety of machinery for finding the formulas: the procedure is just a series of syntactic rewrites. However, we should again warn the reader that the complexity of the separation process (used in both the new proof and the original) and the blow-up in the size of formulas may be non-elementary. Note that there may even be an exponential blow-up in translating from TL to the strict language.

An example transformation is set out in [Reynolds, 2000]. The formula $\phi = qU_{=}p$ rewritten as $p \vee (q \wedge U(p, q))$ in strict form is translated (eventually) to the safety-liveness form, $(FGYP(H\mathbf{false} \wedge p) \wedge GF\mathbf{true}) \vee (FGYP(p \wedge Y(qS(H\mathbf{false} \wedge q))) \wedge GF\mathbf{true})$.

Safety-liveness form and automata

As shown in [Reynolds, 2000], the safety-liveness form allows us to easily and naturally find an equivalent deterministic automaton, i.e. one that accepts

exactly the models of the formula. There are immediate advantages in reasoning tasks if one happens to already have a formula in safety-liveness form. For example, the number of states needed in an equivalent automaton is exponential in the length of the formula: normally, without safety-liveness, a double exponential blow-up is required.

Alternatively, many of the applications of automata to theorem-proving, decision procedures, synthesis of models, and executable temporal logic, can be recast in terms of the properties of formulas in safety-liveness form. There is no need to translate to automata to carry out these reasoning tasks. See [Reynolds, 2000] for details.

Using safety-liveness to axiomatize PCTL*

Perhaps the most useful applications of the safety-liveness form are for those who are developing axiomatizations and theorem-proving methods for extensions of LTL. The important observation here is that automata are increasingly being used in such proofs (see, for example, [Walukiewicz, 1995], [Kesten and Pnueli, 1995], [Kaivola, 1996], and [Reynolds, 2001]) but, as in the last of these references, having an automaton in a completeness proof may involve problems with describing it within the object language. However, past operators and the safety-liveness form allow many of the advantages of automata but stay within the confines of the original language.

Here we mention one such application concerning the widely used branching time logic CTL* from [Emerson and Halpern, 1986]. This logic has been particularly hard to develop reasoning tools for. The relatively recent axiomatization in [Reynolds, 2001] makes great use of automata and needs to rely on a special rule of inference to allow the introduction of new atoms into a proof, which can be used to represent states of an automaton. We end this section by seeing how the safety-liveness form can be used so that if past operators are available then a lot of this extra machinery is not needed. Below we will briefly look at the work in [Reynolds, 2005] where there is a complete axiomatization for the logic PCTL*, CTL* with the past operators: an axiomatization involving only the standard rules of inference and a few dozen simple and intuitive axioms.

It is possible that similar applications may be found in axiomatizations and theorem-proving methods for many of the large number of currently interesting extensions of LTL including branching-time logics, logics with quantified propositions and combinations of temporal and epistemic logics.

PCTL* was defined in [Laroussinie and Schnoebelen, 1994], as the full branching-time computational tree logic CTL* with the addition of (linear) past operators. This was also studied in [Zanardo and Carmo, 1993] and in [Kupferman and Pnueli, 1995]. The latter paper also introduces a different way of adding past operators to CTL* (and CTL): the semantics

of branching past. See [Kupferman and Pnueli, 1995] or [Laroussinie and Schnoebelen, 2000] for details.

CTL* has been hard to axiomatize despite its recognized wide applicability. The problem is seen as its limit closure property: roughly the idea that any increasing sequence of prefixes of paths through a structure is part of one path. The solution in the axiomatization [Reynolds, 2001] is to bring in automata and a related special rule of inference. PCTL*, incorporating CTL*, and all the advantages of past operators, is even more useful but exhibits the same limit closure problem.

The formulas of PCTL* are built from atomic propositions recursively using classical connectives, the LTL+Past temporal connectives Y, $S_=$, T, and $U_=$ and the path-switching modality E: if α and β are formulas then so are $Y\alpha$, $\alpha S_=\beta$, $T\alpha$, $\alpha U_=\beta$, and $E\alpha$. We use the usual LTL+Past abbreviations plus $A\alpha \equiv \neg E\neg\alpha$.

Formulas are evaluated in *(total) Kripke structures*, $M = (S, R, g)$ where:

S is the non-empty set of *states*

R is a total binary relation $\subseteq S \times S$
(i.e. for every $s \in S$, there is some $t \in S$ such that $(s, t) \in R$) and

g $: S \to \wp(\mathcal{L})$ is a labelling of the states with sets of atoms.

A *fullpath* in M is an infinite sequence $b = \langle b_0, b_1, b_2, ...\rangle$ of states of M such that for each i, $(b_i, b_{i+1}) \in R$.

Truth of formulas is evaluated at indices in fullpaths in structures. We write $M, b, i \models \alpha$ iff the formula α is true of the fullpath b at the index (time) i in the structure $M = (S, R, g)$. This is defined recursively by:

$M, b, i \models \mathbf{true}$

$M, b, i \models p$ iff $p \in g(b_i)$, any $p \in \mathcal{L}$

$M, b, i \models T\alpha$ iff $M, b, i + 1 \models \alpha$

$M, b, i \models \alpha U_=\beta$ iff there is some $j \geq i$ such that $M, b, j \models \beta$ and for each k, if $i \leq k < j$ then $M, b, k \models \alpha$

$M, b, i \models Y\alpha$ iff $i > 0$ and $M, b, i - 1 \models \alpha$

$M, b, i \models \alpha S_=\beta$ iff there is some $j \leq i$ such that $M, b, j \models \beta$ and for each k, if $j < k \leq i$ then $M, b, k \models \alpha$

$M, b, i \models E\alpha$ iff there is a fullpath b' such that $\langle b_0, \ldots, b_i\rangle = \langle b'_0, \ldots, b'_i\rangle$ and $M, b', i \models \alpha$

We say that α is valid in PCTL* iff for all Kripke structures M, for all fullpaths b in M, for all indices i, we have $M, b, i \models \alpha$.

To find a Hilbert system capable of deriving exactly these validities, [Reynolds, 2005] extends the axiom system for LTL (given in [Lichtenstein *et al.*, 1985]) by the usual S5 rules and axioms for path-switching, plus that

propositional atoms only depend on states:

APS $p \rightarrow Ap$, for each atomic proposition p

plus some interaction between modalities:

AT $AT\alpha \rightarrow TA\alpha$

AY $AY\alpha \leftrightarrow YA\alpha$

plus the limit closure axiom from [Reynolds, 2001]:

LC $AG(E\alpha \rightarrow ET((E\beta)U(E\alpha))) \rightarrow (E\alpha \rightarrow EG((E\beta)U(E\alpha)))$

THEOREM 13 ([Reynolds, 2005]). *The Hilbert system is sound and complete for PCTL*.*

The completeness proof is very similar to that in [Reynolds, 2001] which shows completeness for a Hilbert system for CTL*. The interesting part is not the addition of the past operators: these can be handled by standard linear temporal logic techniques for past operators given that the axioms virtually define them in terms of the future time operators.

The interesting part of this completeness proof is to show that we do not need to call on the extra, unusual rule (called AA) which is used in [Reynolds, 2001]. Space limitations prevent us from introducing AA properly here. The rule allows new atoms to be brought into a proof provided that their truth values in a tree structure are determined functionally by the truth-values of atoms (both new and original) in the past. There are similarities with the IRR rule of [Gabbay, 1981b]. In using the CTL* proof system to make a derivation and in giving its completeness proof, this special rule allows us to bring a deterministic Rabin automaton into the proof. The new atoms can be used to represent the states of the automaton and the automaton can tell us where we are up to in trying to satisfy LTL formulas along branches of the tree. By making sure certain states do not come up very often if other states do not (using the acceptance pairs of the automaton) we can guarantee that all branches satisfy a particular LTL formula: even the uncountable number of branches which appear in the limit of a construction of such a tree do.

The observation that allows us to modify the completeness proof to cope with PCTL* without the AA rule is that when past operators are available in the language and we have the safety-liveness form of the original formula (and it will be in the original signature) then no new atoms are needed to record our progress along the branches of the tree. We can build a deterministic automaton accepting exactly the models of a given LTL+Past formula by only using (sets or conjunctions of) LTL+Past formulas in the original signature to make up the states.

In fact, with the syntactic method of finding the safety-liveness form there need be no mention of automata at all in the proof. All we have to

do, after translating the formula into the right form, is to make sure that eventually some β_i stays true in our construction (as we build a particular branch from the root towards the leaf) and make sure α_i is true infinitely often.

See [Reynolds, 2005] for details.

4 Separation: future

As we have seen, separation has proven theoretically interesting and both theoretically and practically useful. However, it is far from fully understood, even in the most common situations. We end with some (as far as we know) open problems about it, which may provide work for future researchers. We focus on *complexity* and *succinctness,* which are of some current interest and for which there is relevant recent work. We thank Carsten Lutz for interesting discussions.

4.1 Initial equivalence

We begin with initial equivalence: in $(\mathbb{N}, <)$, at time 0. Recall from section 3.2 that temporal formulas α, β are said to be *initially equivalent* if $(\mathbb{N}, <, h), 0 \models \alpha \leftrightarrow \beta$ for all assignments $h : L \to \wp(\mathbb{N})$.

REMARK 14.

(i) By lemma 10, we know that for any US-formula α, there is a U-formula β that is initially equivalent to α.

(ii) If we combine lemma 10 with the separation algorithm obtained from the proof of theorem 7, we obtain an algorithm to construct β from α, but its complexity appears to be non-elementary. However, [Schnoebelen, 2003, footnote 22] notes that an *elementary upper bound* on the minimal size of an U-formula initially equivalent to a given US-formula can be obtained by combining the standard translation from US-formulas to counter-free Büchi automata and the elementary translation from these automata to U-formulas using results from [Wilke, 1999]. This also gives an *elementary algorithm* to perform the translation.

(iii) [Laroussinie *et al.*, 2002, theorem 3.1] showed that US can be *exponentially more succinct* than U over $(\mathbb{N}, <)$ at time 0. That is, obtaining β from α above can introduce an exponential blow-up in size. Hence, any algorithm to construct β from α requires at least exponential time in the worst case.

This prompts the following questions:

Q1. What is the exact *complexity* of the problem of constructing β from α in remark 14?

Q2. *(Succinctness)* What is the minimum length of β in terms of the length of α?

4.2 Separation

Similar questions can be asked about separation itself. As we said, the proof of theorem 7 provides an algorithm that constructs, for any US-formula α, a US-formula β that is (syntactically) separated and equivalent to α over $(\mathbb{N}, <)$. However, very little is known about the complexity of separation algorithms and the succinctness of their output.

We pose our questions in the general context of a temporal logic $\mathcal{T}$ with the separation property over a class $\mathcal{C}$ of flows of time. Assuming that validity of $\mathcal{T}$-formulas over $\mathcal{C}$ is decidable, it follows from the proof of lemma 9 that there exists an algorithm to separate any $\mathcal{T}$-formula α over $\mathcal{C}$. We just enumerate all $\mathcal{T}$-formulas, stopping when a separated one equivalent to α is found. Separatedness and equivalence are decidable (cf. lemma 9 for the former), so this is an effective process; and since $\mathcal{T}$ has the separation property, it will terminate.

Q3. *Complexity of separation.* What is the optimal complexity of algorithms that, given a $\mathcal{T}$-formula α, output a separated $\mathcal{T}$-formula equivalent to α over $\mathcal{C}$?

This problem has many versions, depending on circumstances. The chief concrete instances of it are

(a) for U, S over $(\mathbb{N}, <)$,

(b) for U, S over $(\mathbb{N}, <)$, but requiring the output formula to be *syntactically* separated,

(c) for U, S over Dedekind-complete time,

(d) for U, S, and the Stavi connectives over linear time.

It follows from remark 14(iii) that over $(\mathbb{N}, <)$, all separation algorithms require at least exponential time.

Q4. *Succinctness of separation.* What can one say about the length of a shortest separated $\mathcal{T}$-formula equivalent over $\mathcal{C}$ to a given $\mathcal{T}$-formula?

This is asking whether there is an inevitable loss of succinctness when separating a formula, and if so, how much? Again, there are various concrete instances of the problem. For U and S over $(\mathbb{N}, <)$, it

follows from remark 14(iii) that separation sometimes incurs an exponential increase in length. It would be interesting if non-separated formulas were non-elementarily more succinct than separated ones in this case, since it follows from [Sistla and Clarke, 1985] that satisfiability over $(\mathbb{N}, <)$ for both arbitrary US-formulas and for U-ones (which are certainly separated) is PSPACE-complete, and both have the exponential-size model property. It would also mark a difference from the special case discussed in remark 14(ii).

REMARK 15. Questions Q3 and Q4 are connected. The following are equivalent:

(a) There is an elementary algorithm to separate US-formulas over $(\mathbb{N}, <)$,

(b) There is an elementary upper bound on the length of a shortest separated US-formula equivalent over $(\mathbb{N}, <)$ to a given formula.

$(a) \Rightarrow (b)$ is trivial, since it takes at least as much time to compute the separated formula as to output it. For $(b) \Rightarrow (a)$, recall from [Sistla and Clarke, 1985] that there is an elementary (PSPACE) algorithm to decide equivalence of US-formulas over $(\mathbb{N}, <)$. So assuming (b) and given a US-formula α, we can enumerate all separated US-formulas of length at most the elementary bound obtained from α, using lemma 9 to verify separatedness, and checking each for equivalence to α. This is an elementary algorithm to separate α.

Obviously, there are more refined results, bounding the complexity of each side in terms of the other.

4.3 Other sets of connectives

Now we consider what might happen when we change the set of connectives.

Consider any temporal logic $\mathcal{T}$ with finitely many connectives which are all first-order-definable. As US is expressively complete over $(\mathbb{N}, <)$, each $\mathcal{T}$-formula α can be translated into a US-formula $\alpha^\dagger$ that is equivalent to α over $(\mathbb{N} <)$. The translation is straightforward. For each n-ary $\mathcal{T}$-connective $\sharp$, we fix a US-formula $\beta_\sharp(p_1, \ldots, p_n)$ that is equivalent over $(\mathbb{N}, <)$ to $\sharp(p_1, \ldots, p_n)$. Such a formula $\beta_\sharp$ exists by expressive completeness of US over $(\mathbb{N}, <)$. Now we put

- $p^\dagger = p$ for any atom p,
- $(\alpha \wedge \beta)^\dagger = \alpha^\dagger \wedge \beta^\dagger$ and $(\neg\alpha)^\dagger = \neg\alpha^\dagger$,
- $\sharp(\alpha_1, \ldots, \alpha_n)^\dagger = \beta_\sharp(\alpha_1^\dagger, \ldots, \alpha_n^\dagger)$.

The translation $\alpha \mapsto \alpha^\dagger$ entails at most an exponential increase in size (and such an increase can be attained). To see this, let $d(\alpha)$ be the depth of nesting of boolean and temporal connectives in the formula α. Formally, $d(p) = 0$ for an atom p, $d(\neg\alpha) = 1 + d(\alpha)$, $d(\alpha \wedge \beta) = 1 + \max(d(\alpha), d(\beta))$, and $d(\sharp(\alpha_1, \ldots, \alpha_n)) = 1 + \max\{d(\alpha_1), \ldots, d(\alpha_n)\}$, for each temporal connective $\sharp$. Here, α can be either a $\mathcal{T}$-formula or a US-formula. Let k be the maximal value of $d(\beta_\sharp)$ (running over all $\mathcal{T}$-connectives $\sharp$). We will assume that $k \geq 1$. Then every nested occurrence of a boolean or temporal connective in α raises the depth of $\alpha^\dagger$ by at most k. So we see that

$$d(\alpha^\dagger) \leq k \cdot d(\alpha) \quad \text{for all } \mathcal{T}\text{-formulas } \alpha,$$

and indeed, this is easily proved formally by induction on α.

Write $|\alpha|$ for the length of α, ignoring commas and brackets for simplicity. By considering formation trees of formulas, we see that for each US-formula β we have $|\beta| \leq 2^{d(\beta)}$. For $\mathcal{T}$-formulas α, we trivially have $d(\alpha) \leq |\alpha|$. So the length $|\alpha^\dagger|$ of the translation is bounded in terms of the length $|\alpha|$ of the original formula by

$$|\alpha^\dagger| \leq 2^{d(\alpha^\dagger)} \leq 2^{k \cdot d(\alpha)} \leq 2^{k|\alpha|}.$$

Clearly, then, the translation $\alpha \mapsto \alpha^\dagger$ is effective and takes at most exponential time. The main requirements were that the connectives in $\mathcal{T}$ are finite in number and have first-order definitions, and that US is expressively complete over $(\mathbb{N}, <)$. So if $\mathcal{T}$ is expressively complete as well, the translation process can be reversed. The translation obviously preserves purity and being separated, since these are defined semantically. It follows that

PROPOSITION 16. *Let $\mathcal{T}$ be a finite set of first-order-definable connectives that is expressively complete over $(\mathbb{N}, <)$. Then the following are equivalent:*

- *There is an elementary algorithm to separate $\mathcal{T}$-formulas over $(\mathbb{N}, <)$.*
- *There is an elementary algorithm that separates US-formulas over $(\mathbb{N}, <)$.*

A similar result holds for succinctness. As in remark 15, the complexity estimates can be refined. Still, they do not give exact bounds, so the following questions are perhaps worth answering:

Q5. Find a finite set $\mathcal{T}$ of connectives that is expressively complete over $(\mathbb{N}, <)$, and such that the complexity of the problem of separating $\mathcal{T}$-formulas is least possible.

Q6. Find a finite set $\mathcal{T}$ of connectives that is expressively complete over $(\mathbb{N}, <)$, and such that the length of the smallest separated $\mathcal{T}$-formula equivalent to a given $\mathcal{T}$-formula is as small as possible.

There are special problems for other flows of time.

Q7. Over linear time, or over $(\mathbb{Q}, <)$:

(a) Is it decidable whether a given US-formula can be separated?

(b) [Rabinovich] Is it decidable whether a given first-order formula $\varphi(x, P_1, \ldots, P_n)$ is equivalent to a US-formula?

In each case, if the answer is positive, one can ask about complexity.

We hope that these questions will find interesting solutions before too long. At any rate, they show that the topic of separation, after more than 20 years, is still very much alive.

BIBLIOGRAPHY

[Amir and Gabbay, 1987] A. Amir and D. M. Gabbay. Preservation of expressive completeness in temporal models. *Information and Computation*, 72:66–83, 1987.

[Amir, 1982a] A. Amir. Expressive completeness failure in branching time structures. *Journal of Computer and System Sciences*, 34(1), 1982.

[Amir, 1982b] A. Amir. *Functional Completeness in Tense Logic*. PhD thesis, Bar-Ilan University, Ramat-Gan, Israel, 1982.

[Amir, 1985] A. Amir. Separation in nonlinear time models. *Information and Control*, 66:177–203, 1985.

[Barringer *et al.*, 1996] H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future*. Research Studies Press, Somerset, 1996.

[Bowman *et al.*, 2003] H. Bowman, H. Cameron, P. King, and S. J. Thompson. Mexitl: Multimedia in executable interval temporal logic. *Formal Methods in System Design*, 22:5–38, January 2003.

[Emerson and Halpern, 1986] E. Emerson and J. Halpern. 'Sometimes' and 'not never' revisited: on branching versus linear time. *J. ACM*, 33, 1986.

[Etessami and Wilke, 2000] K. Etessami and T. Wilke. An until hierarchy and other applications of an Ehrenfeucht–Fraïssé game for temporal logic. *Information and Computation*, 160:88–108, 2000.

[Fagin *et al.*, 1995] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[Finger *et al.*, 1993] M. Finger, M. Fisher, and R. Owens. Metatem at work: Modelling reactive systems using executable temporal logic. In *6th Intl. Conf. on Industrial and Engineering applications of AI and Expert Systems*, 1993.

[Gabbay *et al.*, 1980] D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *7th ACM Symposium on Principles of Programming Languages, Las Vegas*, pages 163–173, 1980.

[Gabbay *et al.*, 1994] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal logic: mathematical foundations and computational aspects, Vol. 1.* Clarendon Press, Oxford, 1994.

[Gabbay, 1981a] D. Gabbay. Expressive functional completeness in tense logic (preliminary report). In U. Monnich, editor, *Aspects of Philosophical Logic*, pages 91–117. Reidel, Dordrecht, 1981.

[Gabbay, 1981b] D. M. Gabbay. An irreflexivity lemma with applications to axiomatizations of conditions on tense frames. In U. Monnich, editor, *Aspects of Philosophical Logic*, pages 67–89. Reidel, Dordrecht, 1981.

[Gabbay, 1989] D. Gabbay. Declarative past and imperative future: Executable temporal logic for interactive systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceedings of Colloquium on Temporal Logic in Specification, Altrincham, 1987*, volume 398 of *Lecture Notes in Computer Science*, pages 67–89. Springer-Verlag, 1989.

[Hodges, 1993] W. Hodges. *Model theory*, volume 42 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 1993.

[Immerman and Kozen, 1987] N. Immerman and D. Kozen. Definability with bounded number of bound variables. In *LICS87, Proceedings of the Symposium on Logic in Computer Science, Ithaca, New York*, pages 236–244, Washington, 1987. Computer Society Press.

[Jonker and Wijngaards, 2003] C. M. Jonker and W. C. A. Wijngaards. A temporal modelling environment for internally grounded beliefs, desires and intentions. Technical Report 040, Utrecht: UU, owi CKI (Artificial intelligence preprint series), 2003.

[Kaivola, 1996] R. Kaivola. Axiomatising extended computation tree logic, in trees in algebra and programming. In *CAAP'96, 21st International Colloquium, Proceedings*, volume 1059, pages 87–101. Springer, 1996.

[Kamp, 1968] H. Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.

[Kesten and Pnueli, 1995] Y. Kesten and A. Pnueli. A complete proof system for QPTL. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 2–12, San Diego, California, 26–29 June 1995. IEEE Computer Society Press.

[Kupferman and Pnueli, 1995] O. Kupferman and A. Pnueli. Once and for all. In *Proc. 10th IEEE Symposium on Logic in Computer Science*, pages 25–35, San Diego, June 1995.

[Laroussinie and Schnoebelen, 1994] F. Laroussinie and Ph. Schnoebelen. A hierarchy of temporal logics with past. In *Proc. STACS'94, Caen, France*, volume 775 of *LNCS*, pages 47–58. Springer–Verlag, 1994.

[Laroussinie and Schnoebelen, 2000] F. Laroussinie and Ph. Schnoebelen. Specification in CTL+Past for verification in CTL. *Information and Computation*, 156:236–263, 2000.

[Laroussinie *et al.*, 2002] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. 17th Annual IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 383–392. IEEE Computer Society Press, 2002. Available at `www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/LMS-lics2002.pdf`.

[Lichtenstein *et al.*, 1985] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In R. Parikh, editor, *Logics of Programs* (*Proc. Conf. Brooklyn USA 1985*), volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer-Verlag, Berlin, 1985.

[Marx, 2004] M. Marx. Conditional XPath, the first order complete XPath dialect. In *Proc. ACM SIGMOD/PODS (Principles of Database Systems)*, 2004. Available at `www.sigmod.org/pods/proc04/`.

[McNaughton and Papert, 1971] R. McNaughton and S. Papert. *Counter Free Automata*. MIT Press, 1971.

[McNaughton, 1966] R. McNaughton. Testing and generating infinite sequences by finite automata. *Information and Control*, 9:521–530, 1966.

[Reynolds, 1999] M. Reynolds. The complexity of temporal logic over the reals. 1999. Submitted.

[Reynolds, 2000] M. Reynolds. More past glories. In *Fifteenth Annual IEEE Symposium on Logic in Computer Science (LICS'2000), Santa Barbara, California, USA, June 26-28, 2000*, pages 229–240. IEEE, 2000.

[Reynolds, 2001] M. Reynolds. An axiomatization of full computation tree logic. *J. Symbolic Logic*, 66(3):1011–1057, 2001.

[Reynolds, 2005] M. Reynolds. An axiomatization of PCTL*. *Information and Computation*, 201:72–119, 2005.

[Schlingloff, 1992] B-H. Schlingloff. Expressive completeness of temporal logic over trees. *Journal of Applied Non-classical Logics*, 2:157–180, 1992.

[Schnoebelen, 2003] Ph. Schnoebelen. The complexity of temporal logic model checking. In *Proc. 4th Workshop on Advances in Modal Logic (AiML'02)*, pages 393–436. King's College Publications, 2003.

[Sistla and Clarke, 1985] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32:733–749, 1985.

[Walukiewicz, 1995] I. Walukiewicz. A complete deductive system for the μ-calculus. Research Series RS-95-6, BRICS, Department of Computer Science, University of Aarhus, January 1995. 39 pp.

[Wilke, 1999] T. Wilke. Classifying discrete temporal properties. In *Proc. 16th Ann. Symp. Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1999.

[Zanardo and Carmo, 1993] A. Zanardo and J. Carmo. Ockhamist computational logic: Past-sensitive necessitation in CTL. *J. Logic Computat.*, 3(3):249–268, June 1993.

Department of Computing
Imperial College London
London SW7 2AZ, U.K.
imh@doc.ic.ac.uk

School of CS & SE
University of Western Australia
Western Australia, 6009
reynolds@csse.uwa.edu.au