Abstract

We introduce a substructural propositional calculus of Sequential Dynamic Logic (SDL) that subsumes a propositional part of dynamic predicate logic (DPL), and is shown to be expressively equivalent to propositional dynamic logic (PDL). Completeness of the calculus with respect to the intended relational semantics is established.

Keywords: dynamic logic, substructural logics, dynamic predicate logic, sequent calculus

1 Introduction

The world we reason about and act in is essentially dynamic, which is the ultimate reason why a logical analysis of the world and information dynamics occupies today an important place in general logical theory. There exist, however, a lot of possibilities and choices about how such a dynamics could be captured in a logical representation, each with advantages and problems of their own. It should be obvious that both an independent and comparative study of such alternatives is essential for a proper analysis and handling of the dynamic reality.

The majority of traditional logical approaches to dynamic evolution, including temporal and dynamic logics, stem from a fundamental modal logic paradigm, according to which the dynamic universum is a set of ‘ordinary’, static worlds structured by inter-world accessibility relations. A syntactic counterpart of this view is that Dynamic Logic, broadly understood, is a conservative extension of (static) classical Boolean logic with (dynamic) modal operators that describe ‘accessible static options’ for each static world. By its very language,
this paradigm presupposes that the essential information about the dynamics of the universe can be expressed, ultimately, in terms of such static accessibility assertions.

There is no need to argue these days that the above modal paradigm has provided (and still constitutes) a powerful and versatile tool for a logical analysis of dynamic discourses. There is something to be said, however, about discrepancies that exist between this modal representation and the ways the dynamic reality is represented in ordinary language discourse.

One of the most salient discrepancies of this kind is that the natural language seems to treat static and dynamic descriptions much on the same footing. In particular, language copulas and expressions corresponding to logical connectives of conjunction, disjunction, implication and negation seem to be freely applied to clauses of both kinds. For example, a sentence ”Event $E$ brings about an effect $A$” on this understanding is invariably expressible as an implication $E \rightarrow A$, whereas in propositional dynamic logic it should be ‘compartmentalized’ as either a modal statement $[E]A$ (if $E$ is an action/program), or as $E \rightarrow A$ when $E$ is a static proposition

Moreover, we argue that the above mentioned discrepancies transcend the boundaries of small, inessential syntactic variations, and provide an initial justification for an alternative logical paradigm according to which a logical description of a dynamic universe could be obtained by a direct generalization of classical logic to a dynamic setting.

Of course, we have to pay for this generalization by sacrificing some of the principles of Boolean logic. It is well known, for instance, that the language copula ‘and’ is not commutative in linguistic discourse - ‘I drove home and drunk a lot’ is dangerously different from ‘I drunk a lot and drove home’. Still, the idea is that it is possible to single out a natural logical core of dynamic reasoning by assigning a more dynamic meaning to the ubiquitous logical connectives in a way that would give us a comprehensive basis for representing the dynamic world.

The above idea is not new, of course. A first precise formalization of this idea has been suggested twenty years ago in the framework of Dynamic Predicate Logic (DPL), described in [Groenendijk and Stokhof, 1991]. The syntax of DPL included the usual logical connectives of conjunction, disjunction, implication and negation, but was based on a semantic interpretation in which propositions were directly represented as relations among worlds. In this way DPL instantiated another important idea, namely that meaning of linguistic expressions should be viewed, in general, as a capability of context change. This dynamic view of meaning occupies today a prominent place in natural language semantics (see, e.g., [van Eijck and Stokhof, 2006]).

It should be noted, however, that our Sequential Dynamic Logic, described below, is purported to capture general dynamic reasoning, so it is not restricted to specific (mainly linguistic) interpretations of DPL (see [Dekker, 2008] for a recent overview of the latter). This logic (and its variations/extensions) is

\footnote{or even as $[E?]A$ with an additional syntactic sugar for tests.}
intended to to be applied to reasoning about actions and change in AI, logic programming, and even to general causal reasoning. The present study, however, has more modest objectives. Namely, it is intended to show that our suggested generalization of classical logic is capable (at least) of reaching the level of sophistication and expressivity of propositional dynamic logic (PDL). Moreover, we hope the reader will see that our alternative representation actually provides a significant simplification of the corresponding formal development.

The plan of the paper is as follows. We introduce first the language of sequential dynamic logic that includes propositional atoms of two kinds, dynamic transitions, or actions, and static atoms, or tests. The syntax of the language will be formed, however, by using the usual connectives of conjunction, disjunction, and negation. The semantics of this language will be given in the framework of labeled transition models, which are Kripke structures in which each propositional atom is interpreted as a binary relation. These models will determine also the intended notion of dynamic inference for our language.

Next we will show that the resulting formalism of sequential dynamic logic is equivalent in expressive power to the modal formalism of (elementary) propositional dynamic logic. This will be established by defining back and forth translations between the two systems.

As a next step, we introduce a sub-structural sequent calculus that provides a logical formalization of SDL, and prove the corresponding completeness theorem.

Finally, we will extend the language of SDL with an iteration connective and thereby obtain a sequential counterpart of the full PDL. We will prove the corresponding completeness result by using (a simplified version of) a filtration method.

2 The Language and Semantics

The vocabulary of sequential dynamic logic (SDL) contains propositional atoms of two kinds, transitions (alias actions, or programs) \( \Pi = \{\pi, \rho, \ldots\} \), and state atoms (also called tests) \( P = \{p, q, \ldots\} \). The language of the logic will be a set of logical formulas \( \{A, B, C, \ldots\} \) constructed from the set of propositional atoms (of either kind) using three propositional connectives, (non-commutative) conjunction \( \land \), (dynamic) negation \( \sim \) and (ordinary) disjunction \( \lor \). Semantic interpretation of these formulas will be given in the framework of a relational semantics.

The following definition describes a variant of a well-known notion of a relational model (see, e.g., [van Benthem, 1996]).

**Definition 2.1.** A transition model is a triple \( M = (S, \Pi \cup P, \mathcal{F}) \), where \( S \) is a set of states, \( \Pi \cup P \) a set of atomic propositions, and \( \mathcal{F} \) a valuation function assigning each atomic proposition a binary relation on \( S \), subject to the constraint that, for any state atom \( p \in P, \mathcal{F}(p) \subseteq Id \) where \( Id = \{(s, s) \mid s \in S\} \).
According to the above description, static propositions can be seen as a particular kind of propositions that possess some specific properties. This difference will be captured in the calculus described later by assigning special structural rules to state propositions.

Propositional atoms are interpreted in a transition model as relations among states. In this sense our language can be viewed as a particular language of (binary) relations, which immediately raises the question what is an appropriate logical language for describing dynamic domains, which is the main subject of this study. The trade-off between expressivity and complexity is especially pressing for relational languages, since 'maximal' choices in expressivity (such as a full Relational Algebra) lead to undecidability.

As a reasonable guidance in the choice of the logical operators in our setting we can make use of the notion of safety for bisimulation (see [van Benthem, 1996]). Roughly, an operation on relations is safe for bisimulation if it preserves the relation of bisimulation between models that are bisimilar with respect to its argument relations. Now, the following Safety Theorem has been proved by Van Benthem (see [van Benthem, 1996, Theorem 5.17]):

**Theorem 2.1** (van Benthem). Any first-order relational operation is safe for bisimulation iff it can be defined using atomic relations and atomic tests, using three operations - composition, dynamic negation and union.

The above three operations correspond, respectively, to the three connectives of our logic, namely conjunction ∧, dynamic negation ∼ and disjunction ∨. These connectives can be interpreted in a transition model as follows:

- \(s, t \models \kappa\) iff \((s, t) \in \mathcal{F}(\kappa)\), for any propositional atom \(\kappa\);
- \(s, t \models A \land B\) iff there exists \(r\) with \(s, r \models A\) and \(r, t \models B\);
- \(s, t \models \neg A\) iff \(s = t\) and there is no \(r\) such that \(s, r \models A\);
- \(s, t \models A \lor B\) iff \(s, t \models A\) or \(s, t \models B\);

The above definitions extend, in effect, the valuation function \(\mathcal{F}\) to all formulas of the language. The following properties of the connectives are immediate from these definitions.

The definition for conjunction ∧ amounts to the equality \(\mathcal{F}(A \land B) = \mathcal{F}(A) \circ \mathcal{F}(B)\), where \(\circ\) is the usual composition of binary relations. This conjunction is of course non-commutative, but it is an associative connective, so we can safely omit parentheses in writing multiple conjunctions. Note also that the conjunction of static propositions collapses to the usual classical conjunction (including commutativity).

The condition for disjunction ∨ amounts to the equality \(\mathcal{F}(A \lor B) = \mathcal{F}(A) \cup \mathcal{F}(B)\). Thus, it is a fully standard disjunction - it is associative, commutative and idempotent.

The dynamic negation ∼ is a most specific connective of our language. Thus, negated proposition is always a subset of \(Id\). In other words, negation determines a static proposition by its very definition, independently of its argument.
This property will be reflected in our calculus by extending the notion of a static proposition to all negated formulas. Note also that, when restricted to static propositions, \( \sim \) behaves exactly as a classical negation. In particular, this implies that, like an intuitionistic negation, the dynamic negation satisfies the Triple Negation Law: \( \sim\sim\sim A \equiv \sim A \).

Speaking more generally, from the semantical point of view, any logical combination of static propositions in our language will be a static proposition. Moreover, restricted to static propositions, the three connectives of our language generate a classical Boolean logic. In this sense the classical logic can be justifiably viewed as a ‘static fragment’ of sequential dynamic logic.

In accordance with this, we will (recursively) define a static proposition as either a static atom, or a negated formula, or else a logical combination of static propositions. A static proposition that is not a logical combination of other static propositions will be called a static literal. Note, however, that a static literal can be an arbitrary complex (negated) formula.

Now we are going to define the dynamic entailment relation determined by a transition model.

For any binary relation \( R \), \( \text{dom}(R) \) will denote its domain, and \( \text{range}(R) \) - its range. In addition, we will canonically extend the valuation function \( F \) to sequences of propositions:

\[
F(A_1 \ldots A_n) = F(A_1) \circ \cdots \circ F(A_n),
\]

where \( \circ \) denotes the composition of binary relations.

The entailment relation of our logic will be based on dynamic inference rules of the form \( a \models A \), where \( A \) is a formula and \( a \) a sequence of formulas. The intended informal meaning of such rules will be taken to be that a processes \( a \) causes an event \( A \).

**Definition 2.2.** A dynamic rule \( a \models A \) will be said to be valid in a relational model \( M \) if \( \text{range}(F(a)) \subseteq \text{dom}(F(A)) \).

\( \models_M \) will denote the set of dynamic rules valid in a relational model \( M \).

Unfolding the above definition, \( a \models A \) is valid in \( M \) if and only if, for any states \( s, t \in S \) such that \((s,t) \in F(a)\), there exists a state \( r \) such that \((t,r) \in F(A)\). This notion of validity corresponds precisely to the inference relation adopted in dynamic predicate logic of [Groenendijk and Stokhof, 1991]².

It should be noted, however, that our language is slightly more expressive than DPL in that it contains a general disjunction connective \( \lor \). In fact, (the propositional part of) DPL can be identified with the sub-language of our language determined by \( \{ \land, \sim \} \). Still, DPL has a ‘static’ counterpart of our disjunction, defined as \( \sim(\sim A \land \sim B) \), and it can be easily verified that the latter coincides with our disjunction on static propositions.

In a few works, DPL has been studied from a logical point of view. Thus, [Hollenberg, 1997] provided a finite equational axiomatization of the corresponding relational algebra (including the extensions with disjunction and tests), while

²It has been called Update-to-Domain Consequence in [van Benthem, 1996].
[Blackburn and Venema, 1995] gave a modal analysis of the corresponding dy-
namic implication (described below). The paper [van Eijck, 1999] is more closely
related to our study, since it provides an axiomatization of Dynamic Predicate
Logic (for the first-order case) in a format very similar to ours (see below).

In what follows, we will occasionally use a few derived connectives definable
in our language:

- **dynamic implication** $A \rightarrow B$, defined as $\sim(A \land \sim B)$.

  A direct semantic definition of dynamic implication is as follows:
  
  $s, t \vDash A \rightarrow B$ iff $s = t$ and for every $r$ with $s, r \vDash A$, there exists $r_0$
  with $r, r_0 \vDash B$;

  Dynamic implication constitutes a propositional counterpart of our dy-
namic inference in the same sense as the classical, material implication
forms a propositional counterpart of classical inference. In other words, it
satisfies the Deduction Theorem:

  $\vDash_M A \rightarrow B$ iff $\vDash_M A \rightarrow B$.

  Consequently, it can be used, if desired, to transform our rule-based se-
nquential logic into a Hilbert-type propositional calculus.

- **Inconsistency** $\bot$, defined as $\sim A \land A$, where $A$ is an arbitrary proposition$^3$.
  As can be seen from the semantic interpretation, $\bot$ corresponds to the
  empty relation $\emptyset$. Note that, since $\emptyset \subseteq Id$, the constant $\bot$ can be safely
  viewed as a static falsity constant.

- **Truth** $\top$, defined as $\sim \bot$. As follows from the definition, $\top$ denotes $Id$, so
  it is an ordinary static truth constant.

### 3 SDL versus Elementary PDL

In this section we are going to show that the language of Sequential Dynamic
Logic is expressively equivalent to the language of Propositional Dynamic Logic.
More precisely, we will establish (polynomial) translations from each of the
languages to another that preserve the respective entailment relations.

Just as the language of our SDL, the language of Propositional Dynamic
Logic$^4$ (see, e.g., [Harel et al., 2000]) is based on two sets of atomic expressions,
*programs* $\Pi = \{\pi, \rho, \ldots\}$ and *tests* $P = \{p, q, \ldots\}$. The language itself, how-
ever involves a construction of two separate kinds of expressions, *formulas* and
*programs*, each with connectives of its own, that are defined by mutual recursion:

**Formulas** Any atomic test is a formula, and if $\phi, \psi$ are formulas, and $\alpha$ is a
program, then $\phi \& \psi$, $\sim \phi$ and $[\alpha] \phi$ are formulas.

$^3$Note that the order is important here, since $A \land \sim A$ may well be consistent.

$^4$Though so far without iteration $^*$ (but see below).
Programs Any atomic program is a program, and if \( \alpha, \beta \) are programs, and \( \phi \) a formula, then \( \alpha; \beta, \alpha \cup \beta \) and \(?\phi\) are programs.

The entailment relation is defined in PDL only for formulas, so the logical properties of programs are established only indirectly by their functioning as parts of formulas.

In order to bring the models of PDL closer to our previous descriptions, we will define them as follows:

**Definition 3.1.** A PDL-model is a quadruple \( M = (S, P \cup \Pi, V, R) \), where \( S \) is a set of states, \( P \cup \Pi \) is a set of atomic expressions, \( V \) is a valuation function assigning each proposition from \( P \) a subset of \( S \), and \( R \) is a function assigning each atomic program \( \alpha \) a binary relation \( R(\alpha) \) on \( S \).

The valuation \( V \) is extended to all formulas using the following definitions:

- \( s \models p \) iff \( s \in V(p) \);
- \( s \models \phi \land \psi \) iff \( s \models \phi \) and \( s \models \psi \);
- \( s \models \neg \phi \) iff \( s \not\models \phi \);
- \( s \models [\alpha]\phi \) iff \( t \models \phi \), for any \( t \) such that \((s, t) \in R(\alpha)\),

where the function \( R \) is extended to all programs as follows:

\[
R(\alpha; \beta) = R(\alpha) \circ R(\beta)
\]
\[
R(\alpha \cup \beta) = R(\alpha) \cup R(\beta)
\]
\[
R(?\phi) = \{(s, s) \mid M, s \models \phi\}
\]

Finally the entailment relation of PDL has the form \( \Gamma \models \phi \), where \( \phi \) is a formula, and \( \Gamma \) a finite set of formulas.

**Definition 3.2.** A rule \( \Gamma \models \phi \) is valid in a model \( M \) (notation \( \Gamma \models^M \phi \)) if for every state \( s \), \( s \models \psi \) for every \( \psi \in \Gamma \) implies \( s \models \phi \).

\( \models^M \) will denote the set of all such rules that are valid in \( M \).

As a first step in a comparison between the two languages, let us note that the respective descriptions of transition models for SDL and PDL are actually notational variants of each other. Indeed, given an SDL-model \( (S, \Pi \cup P, F) \), we can split the valuation function \( F \) into two functions \( R \) and \( V_0 \), obtained by restricting the domain of \( F \) to atomic transitions and state atoms, respectively. Moreover, due to the constraint on \( F \), the function \( V_0 \) is uniquely determined by a function \( V : P \rightarrow \mathcal{P}(S) \) defined as \( V(p) = \{ s \mid (s, s) \in V_0(p) \} \). As a result, we obtain a PDL model \( (S, P \cup \Pi, V, R) \). Conversely, given a PDL-model \( (S, P \cup \Pi, V, R) \), the corresponding SDL-model is obtainable by defining a valuation function \( F \) that coincides with \( R \) on \( \Pi \), while, for every \( p \in P \), \( F(p) \) is taken to be \( \{(s, s) \mid s \in V(p)\} \). Clearly, \( (S, P \cup \Pi, F) \) will be an SDL-model according to our definition.

Taken for granted the above correspondence between SDL-models and PDL-models, we will establish now translations between the two languages that will preserve the respective entailment relations.
From SDL to PDL. To begin with, the following translation transforms any SDL-formula into a program of PDL:

\[
\begin{align*}
\delta(\pi) &= \pi, \\
\delta(p) &= ? p, \\
\delta(A \land B) &= \delta(A); \delta(B), \\
\delta(\neg A) &= ! (\delta(A), \bot) \\
\delta(A \lor B) &= \delta(A) \cup \delta(B)
\end{align*}
\]

The following lemma shows that the translation \(\delta\) is faithful with respect to the respective semantic interpretations in transition models:

**Lemma 3.1.** For any SDL-formula \(A\), and for every model \(M\),

\[F(A) = R(\delta(A)).\]

**Proof.** By an easy induction on the complexity of \(A\). \(\square\)

As a final step, we will adopt a variant of a ‘global’ translation for dynamic inference rules described in [van Benthem, 1996]. A dynamic rule \(A_1 A_2 \ldots A_n \vdash A\) of SDL will correspond to the following formula of PDL

\[[\delta(A_1); \delta(A_2); \ldots; \delta(A_n)](\delta(A)) \uplus\]

The next result shows that this transformation preserves the dynamic inference relation of SDL:

**Theorem 3.2.** \(A_1 A_2 \ldots A_n \vdash_M A\) iff \([\delta(A_1); \delta(A_2); \ldots; \delta(A_n)](\delta(A)) \uplus\) is PDL-valid in \(M\).

**Proof.** The formula \([\delta(A_1); \delta(A_2); \ldots; \delta(A_n)](\delta(A)) \uplus\) is valid in \(M\) iff for any states \(s, t\) such that \((s, t) \in R(\delta(A_1); \delta(A_2); \ldots; \delta(A_n))\), there exists \(r\) such that \((t, r) \in R(\delta(A))\). By the preceding lemma, this is equivalent to the claim that, for any \((s, t) \in F(A_1 \ldots A_n)\) there exists \((t, r) \in F(A)\), which amounts to \(A_1 A_2 \ldots A_n \vdash_M A\). Hence the result. \(\square\)

From PDL to SDL. The reverse translation from PDL to SDL can be performed in one step. The following translation \(\tau\) simultaneously transforms both formulas and programs of PDL to SDL-formulas.

\[
\begin{align*}
\tau(\kappa) &= \kappa, \text{ for any atom } \kappa \in \Pi \cup P \\
\tau(\phi \land \psi) &= \tau(\phi) \land \tau(\psi) \\
\tau(\neg \phi) &= \neg \tau(\phi) \\
\tau(\alpha; \beta) &= \tau(\alpha) \rightarrow \tau(\beta) \\
\tau(\alpha \lor \beta) &= \tau(\alpha) \lor \tau(\beta) \\
\tau(?! \phi) &= \tau(\phi)
\end{align*}
\]

We will prove first the following technical result.
Lemma 3.3. For any PDL-formula $\phi$, any program $\alpha$, and any model $M$,

1. $\tau(\phi)$ is a static proposition, that is, $F(\tau(\phi)) \subseteq Id$.

2. $s \in V(\phi)$ iff $(s, s) \in F(\tau(\phi))$.

3. $R(\alpha) = F(\tau(\alpha))$.

Proof. All claims are proved by simultaneous induction on the complexity of $\phi$ and $\alpha$. The base cases $p \in P$ (for (1) and (2)) and $\pi \in \Pi$ (for (3)) follow directly from the definitions.

For the case $\neg \phi$, (1) follows from the fact that $\neg \tau(\phi)$ is a negated proposition. Moreover, $F(\tau(\neg \phi)) = F(\neg \tau(\phi)) = Id \setminus F(\tau(\phi))$, so by inductive assumption for $\phi$, $V(\neg \phi) = S \setminus V(\phi) = \{s \mid (s, s) \in F(\tau(\neg \phi))\}$.

For the case $\phi \land \psi$, (1) follows from the fact that conjunction ($\land$) of static propositions is a static proposition. Moreover, $F(\tau(\phi \land \psi)) = F(\tau(\phi)) \cap F(\tau(\psi))$ due to the fact that $\tau(\phi)$ and $\tau(\psi)$ are static propositions. This immediately gives (2).

Finally, for the case $[\alpha] \phi$, (1) follows again from the fact that $\tau(\alpha) \to \tau(\phi)$ ($= \neg(\tau(\alpha) \land \neg \tau(\phi)))$ is a negated proposition. Then $(s, s) \in F(\tau([\alpha] \phi))$ iff $(t, t) \in F(\tau(\phi))$, for any $t$ such that $(s, t) \in F(\tau(\alpha))$ (since $\tau(\phi)$ is a static proposition). Now, by the inductive assumption (3), $R(\alpha) = F(\tau(\alpha))$. Therefore $(s, s) \in F(\tau([\alpha] \phi))$ iff $s \in V([\alpha] \phi)$. This proves (2).

To complete the proof, we have to verify (3).

For the case $?\phi$, $F(\tau(? \phi)) = F(\tau(\phi))$, while $s \in V(\phi)$ iff $(s, s) \in F(\tau(\phi))$ by (2). Hence (3) holds.

For the case $\alpha; \beta$, $F(\tau(\alpha; \beta)) = F(\tau(\alpha) \land \tau(\beta))) = F(\tau(\alpha)) \circ F(\tau(\beta))$. By the inductive assumption, the latter is equal to $R(\alpha) \circ R(\beta) = R(\alpha; \beta)$, as required.

Finally, $R(\alpha \cup \beta) = R(\alpha) \cup R(\beta) = F(\tau(\alpha)) \cup F(\tau(\beta)) = F(\tau(\alpha) \land \tau(\beta)) = F(\tau(\alpha \cup \beta))$. This completes the proof.

As a conclusion, the next result shows that the translation $\tau$ preserves the inference relation of PDL.

Theorem 3.4. $\phi_1, \ldots, \phi_n \vdash^pdl_M \psi$ iff $\tau(\phi_1) \ldots \tau(\phi_n) \vdash^p_M \tau(\psi)$.

Proof. Let $\phi$ denote the formula $\phi_1 \& \ldots \& \phi_n \& \neg \psi$. Then $\phi_1, \ldots, \phi_n \vdash^pdl_M \psi$ does not hold iff $\phi$ is consistent iff there exists $s \in V(\phi)$. By (2) of the preceding lemma, $(s, s) \in F(\tau(\phi))$. Moreover,

$$F(\tau(\phi)) = F(\tau(\phi_1) \land \cdots \land \tau(\phi_n) \land \neg \tau(\psi)),$$

where the conjuncts permute because each $\tau(\phi_i)$ is static. By the definition of $F$, $(s, s)$ belongs to this set, for some $s$, iff $\tau(\phi_1) \ldots \tau(\phi_n) \vdash^p_M \tau(\psi)$.

The above two translations show that SDL and PDL are equivalent formal systems already at the level of semantics. In the next section we will proceed to constructing a sequent calculus for SDL.

9
4 Sequential Dynamic Calculus

Small letters $a, b, c, \ldots$ will denote finite sequences of formulas. Such sequences will be called *processes*. As usual, $ab$ will denote the concatenation of sequences $a$ and $b$ (and similarly for $aA$, $aAb$, etc.).

As is customarily in sequent calculi, we will use both ordinary rules of the form $a \vdash A$, and special rules $a \vdash$ with an empty succedent (meaning that $a$ is *inconsistent*). Note, however, that just as in the classical case, such rules are not strictly needed, since they are equivalent to $a \vdash \bot$, where $\bot$ is an arbitrary contradiction. As we mentioned earlier, a formula $\sim A \land A$ can play this role in our case. We will use a common description $a \vdash X$ for rules of both kinds. In other words, $X$ below is either empty, or a single formula.

In the rules below, we will use $\{\phi, \psi, \ldots\}$ to denote static propositions.

A *dynamic consequence relation* is a set of sequents, of the form $a \vdash X$ that satisfies the following rules:

- **General structural rules:**
  \[
  \frac{a \vdash X}{Aa \vdash X} \quad \text{Left Monotonicity} \quad \frac{a \vdash}{aA \vdash} \quad \text{Right Monotonicity}
  \]

- **Structural rules for static propositions:**
  \[
  \frac{\phi \vdash \phi}{\phi \vdash} \quad \text{S-Reflexivity} \quad \frac{ab \vdash X}{a\phi b \vdash X} \quad \frac{a \vdash \phi \quad a\phi b \vdash X}{ab \vdash X} \quad \text{S-Cut}
  \]

- **Rules for negation:**
  \[
  \frac{a \sim A \vdash}{a \vdash A} \quad (\sim L) \quad \frac{a \vdash \sim A}{aA \vdash} \quad (\sim R)
  \]
  (The double line indicates that the rule is valid in both directions.)

- **Rules for conjunction:**
  \[
  \frac{aABb \vdash X}{a A \lor B b \vdash X} \quad (\lor)
  \]

- **Rules for disjunction:**
  \[
  \frac{aAb \vdash X}{a A \lor B b \vdash X} \quad (\lor)
  \]

Already the very form of our sequents involving sequences of propositions (instead of usual sets) indicates that the sequential dynamic logic is substructural, that is, it does not satisfy, in general, the usual structural rules for consequence relations such as contraction, permutation and weakening. Nevertheless, all these structural rules can be shown to hold for static propositions.
Lemma 4.1. If $\phi$, $\psi$ are static propositions, then the following rules hold for dynamic consequence relations:

\[
\frac{a \phi \phi b \vdash X}{a \phi b \vdash X} \quad \text{S-Contraction} \\
\frac{a \phi \psi b \vdash X}{a \psi \phi b \vdash X} \quad \text{S-Permutation}
\]

Proof. (1) $a \phi \phi \vdash \phi$ by S-Reflexivity and Left Monotonicity, so if $a \phi \phi b \vdash X$, then $a \phi b \vdash X$ is derivable by S-Cut.

(2) $\psi \vdash \psi$ by S-Reflexivity, so $\psi \phi \psi \vdash \psi$ by S-Monotonicity. Now if $a \phi \psi b \vdash X$ holds, then $a \psi \phi \psi b \vdash X$ by S-Monotonicity. Together with $a \psi \phi \psi$, this gives precisely $a \psi \phi b \vdash X$ by S-Cut.

Remark. It has been shown in [van Benthem, 1996, Proposition 7.4] that in a sequential setting, the rules S-Reflexivity, S-Monotonicity, S-Cut and S-Contraction, viewed as rules that hold for all propositions, completely determine the structural properties of classical inference. However, in view of the above result, S-Contraction is derivable from the rest of the rules, so our static structural rules are sufficient for determining a ‘classical static sub-inference’ inside a general dynamic inference.

[van Eijck, 1999] contains an axiomatization of the first-order DPL which is closely related to the above formalism. Using our notation, van Eijck’s axiomatization includes Left Monotonicity, S-Reflexivity and S-Permutation, two rules for negation introduction (on left and right), and two rules for double negation elimination (on both sides). It can be easily shown that these rules for negation are equivalent to our rules. In addition, van Eijck’s axiomatization includes ‘first-order’ rules that deal with variables and quantifiers. Two of these rules, however, also have a propositional import, namely Transitivity and Right Conjunction:

\[
\frac{A \vdash B \quad B \vdash C}{A \vdash C} \quad \text{Transitivity} \\
\frac{A \vdash B \quad A \vdash C}{A \vdash B \land C} \quad \text{R-Conjunction}
\]

These rules hold in van Eijck’s system only under certain restrictions on the variables occurring in the relevant formulas. The restrictions are satisfied, however, when $B$ in each rule is a static proposition, in which case the above rules are derivable in our formalism (cf. the derivation of $\land$-Cut below).

We will prove next a few admissible rules and properties of our calculus that will be needed for the completeness theorem.

Lemma 4.2. 1. $\sim A A \vdash$.

2. $a A \vdash$ if and only if $a \sim \sim A \vdash$.

3. For any static proposition $\phi$,

\[
\frac{a \phi b \vdash}{a \phi \sim b \vdash} \quad ab \vdash
\]
Proof. (1) \( \sim A \vdash \sim A \) by S-Reflexivity, so \( \sim A \vdash \sim (\sim A) \).

(2) follows from the combination of \((\sim R)\) and \((\sim L)\) (in both directions).

(3) It is easy to see that conjunction is an associative connective. So assume that we have \( a \land b \vdash \vdash \land \), and let \( B \) be the conjunction of \( b \). Using \((\land)\), it is easy to verify that \( a \land b \vdash a \land B \), while \( a \sim \land b \vdash a \sim \land B \).

Then the following derivation proves \( aB \vdash \vdash \):

\[
\begin{array}{lcl}
\vdash a \sim \land B \vdash a \sim \land \land B \vdash (2) \\
\vdash a \sim \land \land B \vdash S-\text{Permutation} \quad \vdash a \land B \vdash (2) \\
\vdash a \land B \vdash (2) \\
\end{array}
\]

Finally, \( aB \vdash \vdash \) implies \( ab \vdash \vdash \) by a number of (reverse) applications of \((\land)\).

To end this section, let us note that our axiomatization is not fully modular with respect to the three connectives of the language. Notice in this respect that the rules for conjunction and disjunction deal directly only with relevant formulas in the premises of sequents, not in the conclusions. The rest of the properties are derivable, however, using the rules for negation. A good example is the following rule:

\[
\frac{a \vdash A \quad aB \vdash B}{a \vdash A \land B} \quad \land \text{-Cut}
\]

As a matter of fact, the above rule is a logical counterpart of the structural Cut rule involved in the description of a more basic, structural counterpart of our calculus (described in [Bochman and Gabbay, 2010]). Accordingly, this rule cannot be derived using only the rules of conjunction. Still, it is derivable in the full system as follows\(^5\):

\[
\begin{array}{lcl}
\vdash (A \land B) \land B \vdash (1) \\
\vdash a \vdash B \quad \vdash a \vdash (A \land B) \land B \vdash (2) \\
\vdash a \vdash (A \land B) \land B \vdash (2) \\
\vdash a \vdash (A \land B) \land B \vdash (2) \\
\vdash a \vdash (A \land B) \land B \vdash (2) \\
\end{array}
\]

\(^5\)The numbers below refer to derived rules from Lemma 4.2.
4.1 Strong Completeness

In this section we are going to show that dynamic consequence relations are strongly complete for the relational semantics.

To begin with, we have the following

**Lemma 4.3.** If $M$ is a transition model, then $\models_M$ is a dynamic consequence relation.

The proof amounts to a straightforward check of all the rules of a dynamic consequence relation. In order to show that dynamic consequence relations are complete for this semantics, we are going to construct a canonical transition model of a consequence relation.

As a preparation, we will introduce the following important notion:

**Definition 4.1.**
- A trace is finite sequence $T = X_1X_2\ldots X_n$, where each $X_i$ is either a formula, or a (possibly infinite) set of static propositions.
- An instantiation of a trace $T = X_1X_2\ldots X_n$ is a process $x_1x_2\ldots x_n$ such that, for each $i \leq n$, if $X_i$ is a single formula, then $x_i = X_i$, or else $x_i$ is a finite sequence of (static) propositions taken from the set $X_i$.
- A trace $T$ will be said to be inconsistent with respect to a dynamic consequence relation $\models$ (notation $T \models$) if it has an inconsistent instantiation; otherwise it will be called consistent (notation $T \not\models$).

Now, given a dynamic consequence relation $\models$, we will construct a canonical transition model $M_\models = (S_\models, \Pi \cup P, \mathcal{F})$ as follows:

- The set of states $S_\models$ is a set of all maximal consistent sets of static propositions (viewed as one-element traces).
- For any $p \in \Pi \cup P$, $\mathcal{F}(p) = \{(X,Y) \mid X,Y \in S_\models \& XpY \models\}$

We have to verify first that the above construction satisfies the ‘static’ constraint of a transition model.

**Lemma 4.4.** $M_\models$ is a transition model.

**Proof.** Let $\alpha$ be a static atom, and assume that $X\alpha Y \not\models$. Then $X\alpha \not\models$ (note that all instantiations of $X\alpha$ are instantiations of $X\alpha Y$), and therefore $\alpha \in X$, since $X$ is a maximal consistent set of static propositions. Consequently all instantiations of $X\alpha Y$ are instantiations of $XY$, and hence $XY \not\models$. The latter implies that $X \cup Y$ is a consistent trace (by S-Contraction and S-Permutation), and therefore $X = Y$ due to maximality of $X$ and $Y$. Hence $\mathcal{F}(\alpha) \subseteq Id$, as required.

As in classical constructions of canonical models for modal logics, the following lemma shows that the above canonical valuation function can be uniformly extended to all formulas of the language.
Lemma 4.5 (Basic Lemma). In a canonical model $M_\mu$, for any formula $A$ and any states $X, Y \in S_\mu$,

$$X, Y \models A \text{ if and only if } XAY \not\models .$$

Proof. The proof is by induction on the complexity of the formula $A$. The inductive basis holds by the definition of $V$.

Assume that $A = \sim B$. Then $X, Y \models A$ holds iff $X = Y$ and there is no state $Z$ such that $X, Z \models B$. By the inductive assumption, the latter holds iff $X B Z \not\models$, for any $Z$. Now, the latter claim is reducible to $X B \not\models$. Indeed, the direction from right to left is immediate, while if $X B \not\models$, we can apply the usual extension technics (using (3) of Lemma 4.2) in order to extend the trace $X B$ to a consistent trace $X B Z$, for some maximal consistent set $Z$ of static propositions. This will give us the direction from left to right. Hence, we have to show only that $X \sim B Y \not\models$ holds if and only if $X = Y$ and $X B \not\models$.

Suppose first that $X = Y$ and $X B \not\models$. Then $X \sim B \not\models$ by (2) of Lemma 4.2, and therefore $\sim B \in X$ (since $X$ is a maximal consistent set, either $\sim B \in X$, or $\sim B \notin X$). Consequently, we can safely extend $X$ to a consistent trace $X B X$ (due to S-Contraction and S-Permutation, any instantiation of $X \sim B X$ is an instantiation of $X$). Thus, $X \sim B Y \not\models$ holds. In the other direction, if $X \sim B Y \not\models$, then $X \sim B \not\models$, and therefore $\sim B \in X$ (since $X$ is a maximal consistent set). Consequently, $X B \not\models$ by (1) of Lemma 4.2. Moreover, in this case $X Y \models$, which is possible only if $X = Y$ (see the proof of Lemma 4.4). This completes the proof for negation.

Assume now that $A = B \land C$. By definition, $X, Y \models B \land C$ iff there is a state $Z$ such that $X, Z \models B$ and $Z, Y \models C$. By the inductive assumption, this is equivalent, respectively, to $X B \not\models$ and $Z C \not\models$. On the other hand, by the rule $(\land)$ for conjunction, $X (B \land C) Y \models$ is equivalent to $X B C Y \not\models$. Hence it is sufficient to show that $X B C Y \not\models$ holds iff $X B Z \not\models$ and $Z C Y \not\models$, for some state $Z$.

If $X B C Y \not\models$ holds, we can apply (3) of Lemma 4.2 in order to extend the trace $X B C Y$ to a consistent trace $X B Z C Y$, where $Z$ is some maximal consistent set of static propositions. Then the two Monotonicity rules imply both $X B Z \not\models$ and $Z C Y \not\models$, as required. In the other direction, assume that $X B Z \not\models$ and $Z C Y \not\models$, for some state $Z$, but $X B C Y \not\models$. By definition, the latter implies that there exists an inconsistent sequence of formulas of the form $x B C y$, where $x$ contains only formulas from $X$, and $y$ contains only formulas from $Y$. Let $C_0$ denote the conjunction of the sub-sequence $C_y$. Then $x B C_0 \models$ by the rule $(\land)$ for conjunction, and hence $x B \sim C_0 \models$ by (2) of Lemma 4.2. The latter implies that $\sim \sim C_0$ cannot belong to $Z$ (since $X B Z \not\models$), and therefore $\sim C_0 \in Z$. But now $Z C Y \not\models$ implies $\sim C_0 C Y \not\models$, and therefore we can apply $(\land)$ to derive $\sim C_0 C Y \not\models$ - a contradiction with (1) of Lemma 4.2. This completes the case of conjunction.

Finally, assume that $A = B \lor C$. By the inductive assumption, $X, Y \models B \lor C$ iff either $X B Y \not\models$, or $X Y \not\models$. But the rules for disjunction immediately imply that this holds if and only if $X A \lor B Y \not\models$. This completes the proof. \hfill \Box

The Basic Lemma gives us everything we need for proving the main
Theorem 4.6. If $M_\triangleright$ is a canonical transition model of a dynamic consequence relation $\vdash$, then $\vdash = \vdash_{M_\triangleright}$.

Proof. To begin with, for a proposition $A$ and a sequence of propositions $a$, let $A_0$ denote the conjunction of the sequence $a \sim A$. Then it is easy to verify that $a \vdash_{M_\triangleright} A$ does not hold if and only if there exist states $X, Y \in S_\triangleright$ such that $X, Y \models A_0$. By the Basic Lemma, the latter is equivalent to $XA_0Y \not\models \not\triangleright$. By definition, this implies $A_0 \not\triangleright$. Moreover, if $A_0 \not\triangleright$ holds, then (using (3) of Lemma 4.2) $A_0$ can be extended to a consistent trace $XA_0Y$, where $X$ and $Y$ are maximal consistent sets of static propositions. Therefore, $a \vdash_{M_\triangleright} A$ holds if and only if $A_0 \models \not\triangleright$. But $A_0$ is a conjunction of $a \sim A$, so $A_0 \models \not\triangleright$ is equivalent to $a \sim A \models \not\triangleright$ by the rules for conjunction, while the latter is equivalent to $a \models \sim A$ by $(\sim L)$. This completes the proof. □

As a summary, we conclude with the following

Corollary 4.7. $\vdash$ is a dynamic consequence relation if and only if $\vdash = \vdash_{M_\triangleright}$, for some transition model $M_\triangleright$.

The above corollary provides a formal expression for a strong completeness of SDL with respect to transition models. This strong correspondence will no longer hold, however, if we extend the language with an iteration operation described in the next section.

5 Iteration

So far, we have intentionally omitted iteration from our language, and for a number of reasons. First, the iteration operation on programs is of course essential for programming and computation, but it appears less essential for applications outside computation theory. Second, it is not expressible in our formalism already because it is not a bisimulation-safe operation (due to van Benthem’s safety theorem). Moreover, the iteration operation describes transitive closures of relations, which are not first-order definable, and this makes it a difficult operation from a proof-theoretical point of view. Thus, one of the immediate effects of adding iteration to the language of PDL makes the latter a non-compact formalism. More precisely, the corresponding relational semantics makes valid some irreducibly infinite inference rules (cf. [Renardel de Lavalette et al., 2008]). Due to the two-way correspondence between SDL and PDL, established earlier, this will hold also for our formalism.

Despite all said above, we are going to show in this section that SDL can also be extended with iteration in a relatively transparent way, and that the corresponding formalization and completeness proof are no more complex (and often significantly less so) than in PDL.

Semantics and Axiomatization. We extend the language of SDL with a new iteration connective $\ast$ that has the following semantic interpretation in transition models:
• \( s, t \models A^* \) iff there exists a finite sequence of \( A \)-transitions that connects \( s \) with \( t \).

We will use \( SDL^* \) to denote the extended formalism. The corresponding sequential dynamic calculus for \( SDL^* \) can be defined as follows.

A **dynamic consequence relation with iteration** is a dynamic consequence relation that satisfies the following rules for the iteration connective:

\[
\frac{a A^* b \models X}{ab \models X} \quad \frac{a A^* b \models X}{a A a^* b \models X} \quad \phi A \models \phi \quad \frac{\phi A \models \phi}{\phi A^* \models \phi} \quad (\ast I)
\]

\[
\phi A \models \phi \quad \frac{\phi A \models \phi}{\phi A^* \models \phi} \quad (\ast E)
\]

**Remark.** The formalism of \( SDL^* \) turns out to be surprisingly similar to the sequential system \( S \) described in [Kozen and Tiuryn, 2003]. Ignoring some inessential differences concerning the choice of connectives, the only important difference between the formalisms themselves amounts to the fact that the language of \( S \) imposes certain syntactic restrictions on admissible formulas in conclusions of sequents. Roughly, it requires that only static propositions can be conclusions. In this sense, \( SDL^* \) can be seen as a conservative extension of \( S \) that is free of such language restrictions. On the other hand, the restrictions of \( S \) are not as severe as they seem, since any sequent of \( SDL \) can be transformed into an equivalent sequent that satisfies the restrictions of \( S \). We will leave a detailed comparison between \( S \) and \( SDL^* \) to another occasion, but will mention only that the completeness proof for \( S \), given in [Kozen and Tiuryn, 2003], relied on highly nontrivial algebraic results obtained elsewhere. In this respect, the completeness proof for \( SDL^* \), given below, seems to fulfill an explicit request for a direct completeness proof made at the end of their paper.

### 5.1 Filtration and Weak Completeness

The general strategy of proving completeness for SDL with iteration below will be similar to the completeness proof for the ‘full’ PDL given, e.g., in [Kozen and Parikh, 1981] and [Harel et al., 2000]. This strategy can be described as follows.

To begin with, the Basic Lemma 4.5 ceases to hold for the language with iteration. As a consequence, the canonical transition model \( M_b \) constructed earlier cannot serve as a model of the source consequence relation \( \models \), which blocks, in effect, the possibility of proving the corresponding strong completeness theorem. Still, we will prove below that a sequent \( a \models X \) is derivable in SDL from a finite set of sequents \( \Delta \) if and only if any model of \( \Delta \) validates also \( a \models X \). Moreover, this weak completeness claim can be refined by restricting the set of transition models to finite models only. As for PDL, the way of proving this will consist in filtering the model \( M_b \) with respect to a certain finite set of formulas determined by \( \Delta \) and \( a \models X \). In our case, this set is defined as follows:
Definition 5.1. A Fischer-Ladner closure of a set \( Q \) of SDL-formulas is the least set \( FL(Q) \) of formulas containing \( Q \) and such that

- \( FL(Q) \) is closed under subformulas;
- \( FL(Q) \) is closed under single negations;
- If \( (A \land B) \land C \in FL(Q) \) then \( A \land (B \land C) \in FL(Q) \);
- If \( A \in FL(Q) \) then \( A \land \top \in FL(Q) \);
- If \( A^* \land B \in FL(Q) \), then \( A \land (A^* \land B) \in FL(Q) \).
- If \( (A \lor B) \land C \in FL(Q) \), then \( A \land C \in FL(Q) \) and \( B \land C \in FL(Q) \).

Just as for the PDL-language, both the number and the size of the formulas in \( FL(Q) \) are linearly bounded by the size of \( Q \). In particular, if \( Q \) is finite, \( FL(Q) \) is also finite.

Let \( M_b \) be a canonical model of a consequence relation \( \models \), and \( Q \) a finite set of formulas. We define a finite transition model \( M_Q \), called a filtration of \( M \) by \( FL(Q) \), as follows.

Let us say that states \( X \) and \( Y \) from \( M_b \) are equivalent, if \( X \cap FL(Q) = Y \cap FL(Q) \). Note that only static formulas in \( FL(Q) \) are relevant for this relation. For any state \( X \), \( [X] \) will denote the equivalence class containing \( X \). Then we define \( M_Q \) as a transition model \( (S_Q, \Pi \cup P, \mathcal{F}_Q) \), where

- \( S_Q = \{ [X] \mid X \in S \} \);
- \( \mathcal{F}_Q(\kappa) = \{ ([X], [Y]) \mid X\kappa Y \not\models \}, \) for any \( \kappa \in \Pi \cup P \).

Then the following key result can be established.

Lemma 5.1 (Filtration Lemma). For any states \( X, Y \) from \( S_b \),

(i) For any static \( \phi \in FL(Q) \), \( [X], [X] \models_{M_Q} \phi \) iff \( \phi \in X \);

(ii) (a) For any other formula \( A \in FL(Q) \), if \( XAY \not\models \), then \( [X], [Y] \models_{M_Q} A \);
(b) If \( [X], [Y] \models_{M_Q} \top \) and \( \sim(A \land B) \in X \cap FL(Q) \), then \( \sim B \in Y \).

Proof. The proof is by induction on the complexity of formulas from \( L(Q) \).

Case (i) is immediate if \( \phi \) is an atom or a logical combination of static formulas. So we need only to verify the case \( \phi = \sim A \). If \( [X], [X] \models_{M_Q} \sim A \), then, for any state \( Y \), \( [X], [Y] \not\models_{M_Q} A \) and therefore \( XAY \not\models \) by (a). The latter implies \( XA \not\models \), since if \( XA \not\models \), then the trace \( XA \) can be extended to a consistent trace \( XAY \), for some maximal consistent \( Y \). Now, \( XA \models \) implies \( X\sim A \models \), and therefore \( \sim A \in X \), as required. In the other direction, assume that \( \sim A \in X \), but \( [X], [Y] \not\models_{M_Q} A \), for some \( Y \). Then \( \sim(A \land \top) \in X \) (since \( \sim A \models \sim(A \land \top) \)). Moreover, since \( \sim A \in FL(Q) \), we have \( \sim(A \land \top) \in FL(Q) \) by the properties of Fischer-Ladner closure. Therefore \( \top \in Y \) by (b) - a contradiction. This completes the case (i).
Case (a). If \( A \) is an atom, then (a) holds by definition.

Assume that \( B \land C \in L(Q) \), and \( X(B \land C)Y \not\models \). Then \( XBCY \not\models \) by the rule for conjunction, so we can apply (3) of Lemma 4.2 in order to extend the trace \( XBCY \) to a consistent trace \( XBZCY \), where \( Z \) is some maximal consistent set of static propositions. Then the two Monotonicity rules imply both \( XBJY \not\models \) and \( ZCY \not\models \). By the inductive assumption, \( [X], [Z] \models_{M_Q} B \) and \( [Z], [Y] \models_{M_Q} C \), so \( [X], [Y] \models_{M_Q} B \land C \), as required.

Assume that \( B \lor C \in L(Q) \), and \( X(B \lor C)Y \not\models \). Then by the rules for disjunction either \( XBY \not\models \) or \( XCY \not\models \), so by the inductive assumption either \( [X], [Y] \models_{M_Q} B \) or \( [X], [Y] \models_{M_Q} C \), and consequently \([X], [Y] \models_{M_Q} B \lor C \).

Finally, assume that \( A = B^* \). For a state \( Z \), let us consider the set of (static) formulas \( (Z \cap L(Q)) \cup \{ \neg D \mid D \in L(Q) \setminus Z \} \). This set is finite, and it uniquely determines the equivalence class \([Z] \), so if \( \psi \) is a conjunct of this set, then \( W \in [Z] \) iff \( \psi \in W \). Now assume that \( [X], [Y] \models_{M_Q} B^* \), and let \( \phi \) be the formula \( \bigvee \{ \psi \in [Z] \mid [Z], [Y] \not\models_{M_Q} B^* \} \). Clearly, a state \( W \) includes \( \phi \) if and only if \( \psi \in W \). Then \( [W], [Y] \models_{M_Q} B^* \) and also \( [Z], [W] \models_{M_Q} B \) by the inductive assumption (a), so \( [Z], [Y] \models_{M_Q} B^* \) by the semantics of iteration, which contradicts the assumption \( \phi \in Z \). Thus, \( \phi \models \phi \) holds, and hence \( \phi B^\ast \models \phi \) by the rule \((I)\) of iteration, and consequently \( \phi B^\ast \models \phi \). Since \( [X], [Y] \not\models_{M_Q} B^* \), we have \( \phi \in X \) and \( \phi \in Y \) (because \( [Y], [Y] \models_{M_Q} B^* \)), and therefore \( XBCY \models \), as required.

Case (b). If \( \pi \) is an atom, then \( [X], [Y] \models_{M_Q} \pi \) amounts to \( X' \pi Y' \not\models \), for some \( X' \in [X] \) and \( Y' \in [Y] \). Suppose that \( \neg(\pi \land B) \in X \cap FL(Q) \), but \( \neg B \not\in Y \). Then \( \neg(\pi \land B) \in X' \) and \( \neg B \not\in Y' \) (since \( \neg B \in FL(Q) \)). Consequently, \( \neg(\pi \land B) \models Y' \), and therefore \( X' \pi Y' \not\models \) implies \( \neg(\pi \land B) \models \). But \( \neg(\pi \land B) \models \), so \( \neg(\pi \land B) \models \) - a contradiction. This confirms the inductive basis for (b).

Assume that \( [X], [Y] \models_{M_Q} B \land C \), and \( \neg((B \land C) \land D) \in X \cap FL(Q) \). Then there exists a state \( Z \) such that \( [X], [Z] \models_{M_Q} B \) and \( [Z], [Y] \models_{M_Q} C \). But \( \neg((B \land C) \land D) \in X \cap FL(Q) \) by associativity of conjunction and the properties of \( FL(Q) \), so \( [X], [Z] \models_{M_Q} B \) implies \( \neg(C \land D) \in Z \) by the inductive assumption, and therefore \( [Z], [Y] \models_{M_Q} C \) gives \( \neg D \in Y \).

Assume that \( [X], [Y] \models_{M_Q} B \lor C \), and \( \neg((B \lor C) \land D) \in X \cap FL(Q) \). Note that \( \neg((B \lor C) \land D) \models \neg(B \land D) \) and \( \neg((B \lor C) \land D) \models \neg(C \land D) \). Consequently, \( \neg(B \land D) \in X \cap FL(Q) \) and \( \neg(C \land D) \in X \cap FL(Q) \) by the properties of \( FL(Q) \). But we also have that either \( [X], [Y] \models_{M_Q} B \), or \( [X], [Y] \models_{M_Q} C \), so \( \neg D \in X \) by the inductive assumption.

Finally, assume that \( [X], [Y] \models_{M_Q} B^* \) and \( \neg(B^* \land C) \in X \cap FL(Q) \). By the semantic definition, \( [X], [Y] \models_{M_Q} B^* \) implies that either \( [X] = [Y] \), or there is a sequence of states \( X_1, \ldots, X_n = Y \) such that \( [X], [X_1] \models_{M_Q} B \) and \( [X_i], [X_{i+1}] \models_{M_Q} B \), for any \( i < n \). For the first case, we have \( \neg(B^* \land C) \models \neg(B^* \land C) \models \neg \neg C \), which implies \( \neg(B^* \land C) \models \neg \neg C \), and hence \( (B^* \land C) \models \neg C \). Applying \((E)\), we obtain \( \neg(B^* \land C) \models \neg C \), and therefore \( \neg C \in X \). For the second case, \( \neg(B^* \land C) \models \neg \neg C \), which implies \( \neg(B^* \land C) \models \neg \neg C \), and hence \( \neg(B^* \land C) \models \neg C \).
\(~(B \land (B^* \land C))\). Consequently, \(~(B^* \land C) \in X \cap FL(Q)\) implies \(~(B \land (B^* \land C)) \in X \cap FL(Q)\) (by the properties of \(FL(Q)\)), and therefore \(~(B^* \land C) \in X_1\) by the inductive assumption. Applying this step \(n\) times, we conclude that \(~(B^* \land C) \in X_n\), that is, \(~(B^* \land C) \in Y\), and hence \(~C \in Y\) due to the fact that \(~(B^* \land C) \vdash ~C\). This completes the proof. \(\square\)

The above Filtration Lemma provides the main step in proving the corresponding completeness result.

Due to the Horn form of the rules characterizing a dynamic consequence relation, intersections of dynamic consequence relations are again dynamic consequence relations. Consequently, for any set \(\Delta\) of dynamic sequents there exists a least consequence relation containing it. We will denote this consequence relation by \(\vdash_{\Delta}\). If \(\Delta\) is finite, then it can be easily verified that \(\vdash_{\Delta}\) is the set of all sequents derivable from \(\Delta\). In other words, \(a \vdash_{\Delta} A\) holds if and only if \(a \vdash A\) is derivable from \(\Delta\) using the rules for a dynamic consequence relation.

**Theorem 5.2 (Weak Completeness Theorem).** If \(\Delta\) is a finite set of sequents, then \(a \vdash_{\Delta} A\) iff \(a \vdash A\) is valid in every finite transition model that validates \(\Delta\).

**Proof.** Let \(M_{\Delta}\) be the canonical model of \(\vdash_{\Delta}\) (see the description preceding Lemma 4.4). Let \(Q\) denote the following (finite) set of formulas:

\[
Q = \{\land a_i \to A_i \mid a_i \vdash A_i \in \Delta\} \cup \{\land a \to A\},
\]

Finally, let \(M_Q\) be the finite transition model obtained by filtration of \(M_{\Delta}\) by \(L(Q)\).

Assume that \(a \not\vdash_{\Delta} A\). Since any dynamic sequent \(b \vdash B\) is equivalent to \(\vdash \land b \to B\), we have that if \(a_i \vdash A_i \in \Delta\), then \(\land a_i \to A_i\) is included in every state from \(M_{\Delta}\), but \(\land a \to A\) does not belong to at least one state of \(M_{\Delta}\). Now, due to (i) of the Filtration Lemma, the same will hold for the model \(M_Q\), which means that \(M_Q\) validates \(\Delta\), but does not validate \(a \vdash A\). This completes the proof. \(\square\)

6 Summary and Perspectives

In this study, we have shown that a sequential formulation of dynamic logic provides a transparent and convenient representation of dynamic inference. Hopefully, we have persuaded the reader that Dynamic Logic can be consistently viewed as an appropriate generalization of classical logic for dynamic discourses. We also hope that this study will contribute to a better understanding of what is genuine ‘computation’ and what is ‘extraneous mathematics’ in the logical analysis of processes (using van Benthem’s phrase).

Of course, the difference between SDL and traditional dynamic logic should not be overestimated, witness the mutual translations between the two established in the paper. Still, the very possibility or, more precisely, the viability of the non-modal, substructural approach to describing dynamic domains obviously suggests new perspectives and new directions of research. Taking only
one example, the sequential approach provides a different outlook on what can be seen as a basic, connective-free dynamic calculus, which is of interest for many applications beginning with reasoning about actions and change in AI and ending with logic programming. We are intending to explore these issues in the future.

Acknowledgments. We wish to thank the anonymous reviewers for their careful and highly instructive comments.

References

[Blackburn and Venema, 1995] P. Blackburn and Y. Venema. Dynamic squares. *Journal of Philosophical Logic*, 24:469–523, 1995.

[Bochman and Gabbay, 2010] A. Bochman and D. M. Gabbay. Causal dynamic inference. No. 368, 2010. To appear in *Annals of Mathematics and Artificial Intelligence*.

[Dekker, 2008] P.J.E. Dekker. A guide to dynamic semantics. Technical report, ILLC Prepublications, 2008.

[Groenendijk and Stokhof, 1991] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–101, 1991.

[Harel et al., 2000] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Hollenberg, 1997] M. Hollenberg. An equational axiomatization of dynamic negation and relational composition. *J. of Logic, Language and Information*, 6:381–401, 1997.

[Kozen and Parikh, 1981] D. Kozen and R. Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14:113–118, 1981.

[Kozen and Tiuryn, 2003] D. Kozen and J. Tiuryn. Substructural logic and partial correctness. *ACM Trans. Comput. Logic*, 4(3):355–378, 2003.

[Renardel de Lavalette et al., 2008] G. Renardel de Lavalette, B. Kooi, and R. Verbrugge. Strong completeness and limited canonicity for PDL. *Journal of Logic, Language and Information*, 17:69–87, 2008.

[van Benthem, 1996] J. van Benthem. *Exploring Logical Dynamics*. CSLI Publ., 1996.

[van Eijck and Stokhof, 2006] J. van Eijck and M. Stokhof. The gamut of dynamic logics. In Dov M. Gabbay and John Woods, editors, *Handbook of the History of Logic*, volume 7, pages 499–600. North-Holland, 2006.

[van Eijck, 1999] J. van Eijck. Axiomatising dynamic logics for anaphora. *Journal of Language and Computation*, 1:103–126, 1999.