Available online at www.sciencedirect.com
SCIENCE DIRECT®
Annals of Pure and Applied Logic 135 (2005) 1–72
ANNALS OF PURE AND APPLIED LOGIC
www.elsevier.com/locate/apal

Tableaux for constructive concurrent dynamic logic☆

Duminda Wijesekera[a,*], Anil Nerode[b]

[a] *Department of Information and Software Engineering, George Mason University, 160 Sci., Technology Bldg. II, Fairfax, VA 22101, United States*
[b] *Department of Mathematics, Cornell University, Ithaca, NY 14853, United States*

Received 26 June 2001; received in revised form 24 June 2004; accepted 8 December 2004
Available online 16 January 2005

Communicated by S.N. Artemov

Abstract

This is the first paper on constructive concurrent dynamic logic (CCDL). For the first time, either for concurrent or sequential dynamic logic, we give a satisfactory treatment of what statements are forced to be true by partial information about the underlying computer. Dynamic logic was developed by Pratt [V. Pratt, Semantical considerations on Floyd–Hoare logic, in: 17th Annual IEEE Symp. on Found. Comp. Sci., New York, 1976, pp. 109–121, V. Pratt, Applications of modal logic to programming, Studia Logica 39 (1980) 257–274] for nondeterministic sequential programs, and by Peleg [D. Peleg, Concurrent dynamic logic, Journal of the Association for Computing Machinery 34 (2) (1987), D. Peleg, Communication in concurrent dynamic logic, Journal of Computer and System Sciences 35 (1987)] for concurrent programs, for the purpose of proving properties of programs such as correctness. Here we define what it means for a dynamic logic formula to be forced to be true knowing only partial information about the results of assignments and tests. This informal CCDL semantics is formalized by intuitionistic Kripke frames modeling this partial information, and each such frame is interpreted as an idealized concurrent machine (a concurrent transition system). In CCDL, proofs and deductions are ω-height, ω-branching, well-founded labeled subtrees of ω^{ω}. These are a generalization of the signed tableaux of Nerode [A. Nerode, Some lectures

☆ We acknowledge support from the Multiple University Research Initiative "An Integrated Approach to Intelligent Systems", DAAH04-96-1-0341, joint between The University of California at Berkeley (Electronics Research Laboratory), Stanford University (Computer Science Department), and Cornell University (Center for Foundations of Intelligent Systems). This MURI was monitored by the U.S. Army Research Office.
* Corresponding author.
E-mail addresses: duminda@ise.gmu.edu (D. Wijesekera), anil@math.cornell.edu (A. Nerode).

doi:10.1016/j.apal.2004.12.001

in modal logic, Technical Report, M.S.I. Cornell University, 1989, CIME Logic and Computer Science Montecatini Volume, Springer-Verlag Lecture Notes, 1990, A. Nerode, Some lectures in intuitionistic logic, Technical Report, M.S.I. Cornell University, 1988, Marktoberdorf Logic and Computation NATO Summer School Volume, NATO Science Series, 1990 (in press)] stemming from the prefix tableaux of Fitting [M.C. Fitting, Proof Methods for Modal and Intuitionistic Logic, Reidel, 1983]. We demonstrate the correctness of our tableau proofs, define consistency properties, prove that consistency properties yield models, construct systematic tableaux, prove that systematic tableaux yield a consistency property, and conclude that CCDL is complete. This infinitary semantics and proof procedure will be the primary guide for defining, in a sequel, *the* correct finitary CCDL (FCCDL) based on induction principles. FCCDL is suitable for implementation in constructive logic software systems such as Constable's NUPRL or Huet-Coquand's CONSTRUCTIONS. Our goal is to develop a constructive logic programming tool for specification and modular verification of programs in any imperative concurrent language, and for the extraction of concurrent programs from constructive proofs. Subsequent papers will introduce analogous logics for declarative and functional concurrent languages.

1. Introduction

We wish to explore the consequences of partial information about computer and information systems. We choose as our first domain of study of partial information the dynamic logic of concurrent systems. To study this requires a modest acquaintance with concurrency, intuitionistic logic (=*constructive* logic), and dynamic logic. Since there are few who have the background in all three, and the literature in each is fairly opaque, we give the motivation and definitions required from all three areas in Sections 1–3.

The work on program correctness of Floyd [6], Hoare [16], and Burstall [2] inspired Pratt [26,27] to develop a modal logic of non-deterministic sequential programs within full classical logic. There are classic papers by Harel [14,15], and Kozen and Parikh [17], which also use ideas from the algorithmic logic of Engeler and Salwicki; see [14] for references. Excellent introductions can be found in [18,14,15] and [8]. Dynamic logic extends ordinary classical propositional or predicate logic as a language for expressing propositions φ about states of machines, or processes, by introducing formulas of the form $[\alpha]\varphi$, where α is a program. The intended meaning of $[\alpha]\varphi$ is: if a machine is in state s and program α is then executed in any way, then at the termination of the execution of α, φ holds at the new state s' of the machine. In dynamic logic, $\psi \rightarrow [\alpha]\varphi$ expresses Hoare's partial correctness assertion $\psi\{\alpha\}\varphi$, but, unlike Hoare logic, dynamic logic can also express termination of programs by $\neg[\alpha]\neg\varphi$, abbreviated to $< \alpha > \varphi$, within classical logic.

Peleg [23,24] introduced concurrent dynamic logic within classical logic with an intended semantics of a concurrent state transition system and proved a completeness theorem. Here is a definition of a Peleg concurrent transition system. Fix a non-empty set S, whose members are called *states*. Suppose that we are given a collection of *programs* as well. We are given that each program α denotes an *accessibility relation* $R_\alpha \subset S \times P(S)$, where $P(S)$ is the power set of the set of states. The intuition behind $sR_\alpha T$ is that, if we are starting in state s, at the end of the execution of α we are allowed to be in any

state in T, but we retain no knowledge of intermediate states. Through its accessibility relation R_α each program α induces a modal possibility (diamond) operator $< \alpha >$ and a modal necessity (box) operator $[\alpha]$ applying to propositions about states which produces propositions about states. These modal operators are interpreted without reference to any operations on programs to obtain programs, and as a result can be applied flexibly in many situations beside those in dynamic logic.

Concurrent diamond (classical logic) $< \alpha > \varphi$ is true at state s if there is a set $V \subset S$ such that $(s, V) \in R_\alpha$, and for each $v \in V$, φ is true at state v.

Concurrent box (classical logic) $[\alpha]\varphi$ is true at state s if $(s, V) \in R_\alpha$ implies that for all $v \in V$, φ is true at state v.

In the constructive treatment to follow we will replace the notion of *complete state of knowledge about the machine* by a notion of *partial state of knowledge about the machine* in a model in a Kripke frame. The definition of state and the interpretation of diamond will remain formally the same as that in classical dynamic logic, but there will be a change in the definition of box.

To get a dynamic logic corresponding to a machine model and the programs that run on it, one has to describe what the intended atomic programs are, and how complex programs are built up from simple programs by program constructs. For dynamic logic as it has been developed in the past, one envisages the machine as being described by memory locations named by program variables. The state s of the machine is supposed to be determined by the simultaneous contents of all memory locations. A *program variable* x_i is associated with each memory location i. The state s of the machine is identified with a map s of which the domain is the set X of all program variables x_i, $s(x_i)$ being interpreted as the content of the ith memory location. The class of atomic programs is the class of sequential assignments $(x_i \leftarrow t)$, where t is a *program term* built up from program variables and function symbols and individual constants from the language of the relational system. This assignment, applied as a program when the state is s_1, yields a new state s_2, where s_2 coincides with s_1 except possibly for the argument x_i, where $s_2(x_i)$ is defined as $s_1(t)$. This s_2 is conventionally denoted by $s_1[s_1(t)/x_i]$. Then the accessibility relations corresponding to assignments and other standard programming constructs are given below.

Sequential assignment $(x_i \leftarrow t)$ denotes $R(x_i \leftarrow t)$, the set of all pairs $(s_1, \{s_2\})$ such that $s_2 = s_1[s_1(t)/x_i]$, where t is a program term and x_i is a program variable.

Composition $(\alpha; \beta)$ denotes $R(\alpha; \beta)$, the set of all pairs (s, U) such that there is a subset V with $(s, V) \in R(\alpha)$, and for each $v \in V$, there is a subset $U_v \subset U$ with $(v, U_v) \in R(\beta)$, and $U = \cup\{U_v : v \in V\}$.

Iteration Define $R(\alpha^n)$ inductively as

1. $R(\alpha^0)$ is $\{(s, \{s\}) : s \text{ is a state}\}$,
2. $R(\alpha^{n+1})$ is $R(\alpha; \alpha^n)$.

(α^*) denotes $R(\alpha^*) = \cup_{i=1}^{\infty} R(\alpha^i)$.

Non-deterministic choice $(\alpha \cup \beta)$ denotes $R(\alpha \cup \beta) = R(\alpha) \cup R(\beta)$.

Test $(\varphi?)$ denotes $R(\varphi?) = \{(s, \{s\}) : \varphi \text{ holds at } s\}$.

For concurrency, one has to add an extra program construct, *parallel and*, denoted by $\cap$. This notion was first introduced in complexity theory by Chandra, Kozen, and Stockmeyer [3], and later introduced into dynamic logic by Peleg [23]. $\alpha \cap \beta$ is supposed to mirror the fact that α and β are executed simultaneously, starting from a common state w, and is the dual of $\cup$. The intention is that α, β, executed in a state s, lead to a set of states reached by executing α and β in parallel with no interleaving. *Parallel and* $(\alpha \cap \beta)$ denotes $R(\alpha \cap \beta) = \{(s, U \cup V) : (s, U) \in R(\alpha) \text{ and } (s, V) \in R(\beta)\}$.

Classical sequential dynamic logic.

With each concurrent transition system T and its $R'_\alpha s$ we can associate a transition system T^s, the sequential skeleton of T, differing from T in that the corresponding accessibility R'_α for T^s consists of all pairs (s, T) in R_α such that T has at most one element. In the case where T and its sequential skeleton T^s coincide, we call T a sequential transition system, and this is the original Pratt–Harel–Kozen case, with a change in notation. To get their notation back, define the sequential accessibility relation $S_\alpha \subset S \times S$ by $(s, t) \in S_\alpha$ iff $(s, \{t\}) \in R_\alpha$, and the definitions then reduce to

Sequential diamond $< \alpha > \varphi$ is true at state s iff there is a $u \in S$ with $sS_\alpha u$ and φ holds at u.

Sequential box $[\alpha]\varphi$ is true at state s iff for any $u \in S$ with $sS_\alpha u$, φ holds at u.

See [31] for further information.

2. Why develop a constructive theory?

It is desirable to have a logic which allows one to deduce consequences of partial information about the system. Our logic is designed for this purpose. Partial information is expressed as information about domains of definition of function symbols and information about which atomic relations hold in the worlds in Kripke frames. See [20] for the philosophy behind this, or note Scott's recent recommendations for reworking computer science problems in intuitionistic logic. See [22,25] for important previous work.

The Karlsruhe school [10–13,28,29] have implemented finitary classical sequential dynamic logic as software, necessarily based on induction schema replacing infinitary rules. They have developed a sequential dynamic logic programming environment (KIV) for verification of sequential programs. The present paper defines the syntax and semantics which can serve as the foundation for a finitary constructive concurrent dynamic logic based on induction schema replacing infinitary rules. The latter is intended for implementation in a constructive logic system such as Constable's NUPRL as a tool for concurrent program development and verification, and even extraction of concurrent programs from constructive proofs by the *propositions as types* paradigm.

A full treatment of the intuitionistic case is more complicated than appears from the previous literature. Here are two issues which we believe we have resolved in this paper.

- If $< \alpha > \varphi$ is defined as $\neg[\alpha]\neg\varphi$, then proving $< \alpha > \varphi$ merely says that the assumption $[\alpha]\neg\varphi$ leads to a contradiction. It does not supply the usual constructive

reading of the definition of *diamond* given above. We wish to compute the state transition asserted to exist from the evidence supplied. Defining $< \alpha > \varphi$ as $\neg[\alpha]\neg\varphi$ as in the system of Nishimura [22] is a carry-over of classical habits, and precludes uniformly constructing state transitions verifying $< \alpha > \varphi$ after $< \alpha > \varphi$ has been proved. In [19], which we regard as the main precursor to this paper, only $< \alpha >$ is primitive. This suffers from a dual defect, which he recognizes. He suggests developing an independent box, which we have done here. If $[\alpha]\varphi$ expresses termination and is defined as $\neg < \alpha > \neg\varphi$, then we do not get the usual natural constructive reading of termination from the definition of *box* given earlier. In the intuitionistic concurrent dynamic logic that we develop, $< \alpha > \varphi$ is not definable in terms of $[\alpha]\varphi$, nor conversely. In our systems $< \alpha >$ is primitive, as well as $[\alpha]$. Finally we remark that Bozic and Dozen [1,4] developed an intuitionistic modal logic with independent box and diamond for philosophical reasons unconnected with computer science. A yet more general intuitionistic modal logic, which covers all potential computer science applications that we know about, is developed by Wijesekera in [31].

- Previous systems [22,25], and many others in the bibliography, implicitly contain the axiom that $< \alpha >$ distributes over $\vee$. This is not justified in the concurrent case, if we maintain the freedom to assert arbitrary propositions about states. Suppose that there is a U such that $sR_\alpha U$ holds and U has at least two elements. We are free to construct propositions φ, ψ so that φ holds at some element of U and ψ holds at the others and neither holds anywhere else. By assumption, for all $u \in U, \varphi \vee \psi$ holds at u. Since $sR_\alpha U$, it follows that $< \alpha > (\varphi \vee \psi)$ holds at s. By the distributive law, either $< \alpha > \varphi$ holds at s or $< \alpha > \psi$ holds at s. This says that there is a $U^1 \subset U$ which is either U or one element of U only, such that $sR_\alpha U^1$ and for all $u \in U^1$, φ holds at u, or for all $u \in U^1$, ψ holds at u. We can repeat the process on U^1 to get U^2, etc. Finally, if the original U is finite, we get a singleton $\{u\} \subset U$ such that $sR_\alpha\{u\}$. So in the case where the set of states S is finite, the sequential skeleton of the given concurrent system has the same $< \alpha > \varphi$ operator as the original system, but the *concurrency* is absent. In the case where $[\alpha]\varphi$ is defined, as in previous systems, as $\neg < \alpha > \neg\varphi$, we get other unwanted sequentialities. We get the distributive law naturally only when, for all s and all T, $sR_\alpha T$ implies that T has at most one element, that is, in the original non-deterministic sequential case.

The reason that previous systems contained this distributivity axiom is probably that with a conventional definition of Kripke frame, this axiom makes it easy to give a completeness proof using essentially non-deterministic sequential machines. We make do without this distributive law or any substitute and obtain completeness theorems anyway using (non-deterministic) concurrent machines.

3. Motivating partial states

In classical logic, the relational system for the values of states in dynamic logic is completely given in advance. We know for any atomic statement about the relational system whether that statement is true or false in advance. We are allowed to use the values of arbitrary terms and the truth of arbitrary statements instantly in all programs.

1. We assume that as soon as the state s_1 is given as a function on program variables then, for any program term t, the value $s_1(t)$ is instantly available for use in an assignment program resulting in state $s_2 = s_1[s_1(t)/x_i]$. But the new state s_2 depends on $s_1(t)$, which must be computed by following the inductive definition of the program term t. At any stage of an actual computation, we have computed values $s_1(t)$ for some terms only. In effect, we are prepared to make some assignments only.
2. We assume that as soon as the state s_1 is given, we can instantly test whether φ holds at $s_1 : (\varphi?)$, and can use the answer for determining the next state of the computation. But at any finite stage of the computation, we have only computed the validity of some φ.

Our partial states represent partial knowledge of the underlying relational system: as to what atomic relations hold already and as to where functions are already defined. Leaving out reference to time by using partial orderings and using Kripke frames, this is a firm intuitive ground upon which to erect a formal theory. It gives rise to constructive concurrent dynamic logic (CCDL) based on

- concurrent state transition systems,
- partial states of knowledge about the underlying relational system,
- increasing partial states.

One gets theories with different notations and expressiveness depending on the exact logic used and the exact definition of *partial state*. The constructive non-modal part of our intended semantics is a Kripke frame, a partially ordered set of relational systems, which get larger as we go up the ordering. What is the intended denotation in such a relational system for variables, function symbols, and terms? We assume that on each Kripke frame relational system

- individual constants denote elements of the system,
- individual variables range over the whole system,
- function symbols denote partial functions on the system,
- relation symbols denote relations on the system, and
- the denotations of relations and function symbols agree as we go up to larger systems in the Kripke frame.

We use ordinary intuitionistic first order logic as the underlying non-modal language. In our intended semantics for dynamic logic we assume that we have complete knowledge of the assignment of values in the relational system to constants and program variables, but incomplete knowledge of values of functions and therefore incomplete knowledge of values of program terms. The *program terms* are built out of individual constants, program variables, and function symbols which will denote partial functions. We have two disjoint classes of individual variables in the first order formulas, *logic variables* and *program variables*. Occurrences of logic variables *may* be bound; occurrences of program variables *may never* be bound and are *always free*. Here are the different types of terms that we use.

1. Logic terms are built up from individual constants, function symbols, and logic variables.

2. Program terms are built up from constants, function symbols, and program variables.
3. Mixed terms are built up from constants, function symbols, logic variables, program variables.

Of course the mixed terms include both logic terms and the program terms. So the logic terms and the program terms will share use of the same constants and function symbols but have disjoint variables. We only allow substitution of program terms for program variables. (Recall that values assigned to program variables serve solely as contents of memory locations.)

Notice that our set-up of having two different kinds of variables is not absolutely necessary. If we had only one kind of variable (which corresponds to our logic variables) we could have still defined partial states with a partial ordering among them, reflecting the same ordering that is present in the definition of an ordinary Kripke model for first order intuitionistic logic. In a sequel to the present paper we will demonstrate infinitary completeness with respect to that set-up.

4. Syntax and semantics

We introduce the syntax and semantics of constructive concurrent dynamic logic.

4.1. Syntax

Assume a first order language for intuitionistic logic with the following notational conventions.

1. Let $Y = \{y_0, y_1, \dots\}$ be the set of logic variables.
2. Let $C = \{c_0, c_1, \dots\}$ be the set constants.
3. Let $X = \{x_0, x_1, \dots\}$ be the set of program variables.
4. Let $A = \{A_0, A_1, \dots\}$ be the set of predicate letters of various arities.
5. Let $=$ be the equality predicate letter.
6. Let $G = \{g_0, g_1, \dots\}$ be the set of function symbols of various arities.

Here is the notation for terms, logical connectives, atomic programs, and program operators.

Program terms Let $t_0, t_1, \dots$ be all terms built up from constants, program variables, and function symbols.

Logic terms Let $s_0, s_1, \dots$ be all terms built from constants, logic variables, and function symbols.

Mixed terms Let $u_0, u_1, \dots$ be all terms built from constants, logic and program variables, and function symbols.

Atomic programs (**Assignment statements**) $(x_i \leftarrow t_j)$ such that $i, j \in \omega$, where t_j is a program term.

Logical connectives (for building logical formulas) $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies), $\neg$ (not), $\leftrightarrow$ (iff), $\exists$ (there exists), $\forall$ (for all).

Program operators (for building up programs) $\cup$ (non-deterministic choice), $\cap$ (parallel and), ; (composition), * (iteration).

Mixed operators ? (test), $\Diamond$ (possibility), $\Box$ (necessity).

4.1.1. Programs and formulas

Here is the inductive definition of programs and formulas.

Definition 4.1.1. 1. Atomic programs are programs.

If P is a predicate letter of arity n and $t_1, t_2, \ldots, t_n$ are mixed terms, then $P(t_1, t_2, \ldots, t_n)$ is a formula.

2. If α, β are programs and φ, ψ are formulas and y is a logic variable, then
   - $(\alpha^*), (\alpha; \beta), (\alpha \cap \beta), (\alpha \cup \beta)$ are programs.
   - If φ has no free logic variables, no $\Box$, no $\Diamond$ then $(\varphi?)$ is a program. (Note that free program variables are explicitly allowed in φ.)
   - $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi), (\varphi \leftrightarrow \psi), (\neg\varphi), ((\exists y)\varphi(y)), ((\forall y)\varphi(y)), (< \alpha > \varphi), ([\alpha]\varphi)$ are formulas.

Parentheses are omitted at will for readability.

4.2. Semantics

4.2.1. Kripke models for intuitionistic logic: a review

In this subsection we review the definitions of Kripke models for intuitionistic predicate logic with partial terms. We follow the Troelstra–Van Dalen [30] conventions in treating Kripke frames for first order intuitionistic logic. See [30] for details.

Definition 4.2.1 (*Kripke Frames for Intuitionistic Logic*). A frame $((K, \leq), \{(D(k), =_k, \{A_{i,k} : i \in \omega\}, \{g_{j,k} : j \in \omega\}) : k \in K\})$, consists of

1. $(K, \leq)$, a partially ordered set of *possible worlds.*
2. D, a function assigning to each $k \in K$ a non-empty set $D(k)$.
   This is the domain of the possible world k.
3. $=_k$, an equivalence relation on $D(k)$. This is the identity relation denoting $=$ in possible world k.
4. $A_{i,k}$, a relation on $D(k)$ of the appropriate arity. This is the relation denoting A_i in world k.
5. $g_{i,k}$, a partial function on $D(k)$ of the appropriate arity. This is the partial function denoting g_i in world k.

We assume the following on frames.

- D is monotone; that is, for all $k, k' \in K, k \leq k'$ implies $D(k) \subset D(k')$, with $C \subset D(k)$. This says that individuals persist in larger worlds.
- If $d =_k d'$ and $A_{i,k}(d)$, then $A_{i,k}(d')$. This says that equality respects relations in every possible world.
- If $d =_k d'$ and $g_{i,k}(d)$ is defined, then $g_{i,k}(d')$ is defined and $g_{i,k}(d) =_k g_{i,k}(d')$. This says that equality respects partial functions in every possible world.
- If $k \leq k'$ and $d =_k d'$, then $d =_{k'} d'$. This says that equality persists in larger possible worlds.
- If $k \leq k'$, then $A_{i,k} \subset A_{i,k'}$ and $\text{graph}(g_{i,k}) \subset \text{graph}(g_{i,k'})$. This says that partial functions persist in larger possible worlds.

Variable-free terms are interpreted in each $D(k)$ as follows:

Definition 4.2.2. Any $c \in C$ is denoted in $D(k)$ by itself and any $d \in D(k)$ is denoted in $D(k)$ by itself. If g is a n-ary function symbol denoted by g_k in $D(k)$ and v_i for $i = 1, \ldots, n$ is already denoted by $w_i \in D(k)$ and $g_k(w_1, \ldots, w_n)$ is defined in $D(k)$, then $g(v_1, \ldots, v_n)$ is denoted in $D(k)$ by $g_k(w_1, \ldots, w_n)$.

Let the first order language be extended to have a name for each element of $D = \cup\{D(k) : k \in K\}$. The forcing relation $k \vdash \varphi$ between possible worlds k and formulas is usually defined by the following steps when $\Box$ and $\Diamond$ are not present. This is a standard definition in intuitionistic logic.

Definition 4.2.3 (*Forcing in Intuitionistic Logic*). $k \vdash \varphi$ is defined by the following inductive definition for $k \in K$ and φ a sentence in first order logic.

Base case Forcing for Atomic Predicates.

- $k \vdash A_i(v_1, \ldots, v_n)$ if $v_1, \ldots, v_n$ are constant terms and all v_i are defined in k and as $\bar{v}_i$, and $A_{i,k}(\bar{v}_1, \ldots, \bar{v}_n)$.
- $k \vdash (v_1 = v_2)$ if v_1 and v_2 are constant terms defined as $\bar{v}_1$, $\bar{v}_2$ in k and $\bar{v}_1 =_k \bar{v}_2$.

Inductive case Forcing for Non-Atomic Sentences.

- $k \vdash \varphi \wedge \psi$ if $k \vdash \varphi$ and $k \vdash \psi$.
- $k \vdash \varphi \vee \psi$ if $k \vdash \varphi$ or $k \vdash \psi$.
- $k \vdash \varphi \rightarrow \psi$ if, for all $k' \geq k$, $k' \vdash \varphi$ implies $k' \vdash \psi$.
- $k \vdash \forall y \varphi(y)$ if, for all $k' \geq k$ and for all $d \in D(k')$, $k' \vdash \varphi(d)$.
- $k \vdash \exists y \varphi(y)$ if, for some $d \in D(k)$, $k \vdash \varphi(d)$.
- $k \vdash \neg\varphi$ if, for all $k' \geq k$, $(k' \nvdash \varphi)$.

4.2.2. Partial states

In this subsection we assume that we are presented with a conventional Kripke model for first order intuitionistic logic and we define *partial states* and the notion of forcing CCDL formulas by these partial states. The previous subsection was a review of conventional Kripke models for pure intuitionistic predicate logic where forcing was defined, for first order formulas, by *worlds* in the Kripke model.

Definition 4.2.4 (*Partial States*). A partial state of a first order Kripke frame is an assignment of values to program variables in a single possible world $D(k)$. That is, a partial state is a function $s : X \rightarrow D(k) \in K$.[1]

A state formula φ is a formula having no free logic variables. If $x_1, \ldots, x_n$ includes all the program variables of a state formula φ, we write $\varphi(x_1, \ldots, x_n)$.

Remarks. State formulas express properties of complete states of machines. The reference to the codomain k of s is what makes this a *partial state*. We always explicitly mention the codomain k of s. This is essential for a clear understanding of accessibility. Now we extend

[1] Let $S(K)$ be the set of all partial states.

s to a corresponding (partial) map $s : T \to D(k)$, where T is the set of all program terms, by means of the following inductive definition.

Definition 4.2.5 (*Interpreting Terms in Partial States*). Partial terms are interpreted inductively as

Base step $s(c) = c$ for all constants c.

Induction step If t is $g_i(t_1, \ldots, t_n)$ then $s(t)$ is defined iff $s(t_1), \ldots, s(t_n)$ are defined, and then $s(g_i(t_1, \ldots, t_n)) = g_{i,k}(s(t_1), \ldots, s(t_n))$, provided that the term on the right hand side exists in k.

Now we see exactly where the partiality of partial states comes in. A program term t denotes an object in $D(k)$ relative to $s : X \to D(k)$ iff s is defined on t, and then t denotes $s(t)$. Only some program terms will be evaluable by s in $D(k)$, even though all the program variables are evaluable, since the domain of s is the set X of program variables. To repeat, only some evaluations of program terms, and therefore only some assignment programs, can be carried out in world k; the ones that can be carried out grow as the world gets larger. Further, knowledge of the atomic relations that these terms satisfy, and therefore tests that can be performed in the world, grow as we go on to larger worlds.

Definition 4.2.6 (*Ordering Partial States*). Let K be a Kripke frame. Partially order $S(K)$ by $s_1 \leq s_2$ iff $s_1 : T \to D(k)$, $s_2 : T \to D(k')$ with $k \leq k'$ and s_1 is the restriction of s_2 to $D(k)$.

The following semantics is a generalization of Tarski's definition of satisfaction in Kripke frames by means of maps from the set of individual variables to the domain of the relational system, often called satisfaction sequences, where the terminology of satisfaction sequences is replaced by the terminology of partial states. The classical Tarski definition in model theory is adjusted to the possible worlds of a Kripke frame. This definition requires a double induction because of the presence of modal connectives $\Diamond, \Box$ themselves having inductive definitions for their accessibility relations. The definition of forcing requires that we enlarge the original language by adding names of $\cup\{D(k) : k \in K\}$ to the set of constants C.

Definition 4.2.7 (*Kripke Models for CCDL*). The Kripke model (of intuitionistic concurrent dynamic logic) associated with a first order Kripke frame K (for ordinary intuitionistic first order logic) consists of K and a concurrent transition system (with domain the set $S(K)$ of all partial states of K) is given by the simultaneous inductive definition below.

Base step (Atomic Programs and Formulas)

1. $s \vdash A_i(t_1, \ldots, t_n)$ if $s(t_1), \ldots, s(t_n)$ are defined in k, and $(s(t_1), \ldots, s(t_n)) \in A_{i,k}$.[2]
2. (Equality) $s \vdash (t = t')$ if $s(t), s(t')$ are defined in k and $s(t) =_k s(t')$.[3]
3. (Assignment) $R(x_i \leftarrow t)$ is the set of all $(s_1, \{s_2\})$ such that $s_1(t)$ is defined in k and $s_2 = s_1[s_1(t)/x_i]$.[4]

[2] Here A_i is a predicate letter of arity n and $t_1, \ldots, t_n$ are program terms and $s : X \to D(k)$ is a partial state.

[3] Here t, t' are program terms.

[4] $s_2 : X \to D(k)$ is a state.

Induction step (Non-Atomic Programs and Formulas)

1. $s \vdash (\varphi \wedge \theta)$ if $s \vdash \varphi$ and $s \vdash \theta$.
2. $s \vdash (\varphi \vee \theta)$ if $s \vdash \varphi$ or $s \vdash \theta$.
3. $s \vdash (\varphi \rightarrow \theta)$ if, for all $s' \geq s$, if $s' \vdash \varphi$, then $s' \vdash \theta$.
4. $s \vdash (\forall y \varphi(y))$ if, for all s' with s'[5] $\geq s$, and all $d \in D(k')$, we have $s \vdash \varphi(d)$.
5. $s \vdash (\exists y \varphi(y))$ if there is a $d \in D(k)$ such that $s \vdash \varphi(d)$.
6. $s \vdash (\neg\varphi)$ if, for all $s' \geq s$, $s' \nvdash \varphi$.
7. $s \vdash (< \alpha > \varphi)$ if there is an $S' \subset S$ such that $(s, S') \in R(\alpha)$ with, for all $s' \in S'$, $s' \vdash \varphi$.
8. $s \vdash ([\alpha]\varphi)$ if for all $s' \in S$ such that $s' \geq s$, and for all $S' \subset S$ such that $(s', S') \epsilon R(\alpha)$, $s'' \vdash \varphi$ for all $s'' \in S'$.[6]
9. $R(\alpha; \beta)$ is the set of all (s, U) such that there exists a $V \subset S$ with $(s, V) \in R(\alpha)$ and a function defined on V with values subsets $U_v \subset U$ such that for all $v \in V$, $(v, U_v) \in R(\beta)$ and $U = \cup\{U_v : v \in V\}$.
10. $R(\alpha^*) = \cup_{i=1}^{\infty} R(\alpha^i)$.
11. $R(\varphi?) = \{(s, \{s\}) : s \vdash \varphi\}$.
12. $R(\alpha \cup \beta) = R(\alpha) \cup R(\beta)$.
13. $R(\alpha \cap \beta) = \{(s, U \cup V) : (s, U) \in R(\alpha) \text{ and } (s, V) \in R(\beta)\}$.

Remark. The definition of $*$ that we use here is not the same as that Peleg [24] uses in his work on classical concurrent dynamic logic. He points out that this definition restricts the different branches of program $\alpha \cap \beta$ to having the same length. Nevertheless his $*$ is definable from this $*$ and the other program constructs of $\cap$, $\cup$, and sequential composition. Our definition of accessibility for α^*, $R(\alpha^*)$ is chosen for its greater constructive content. This definition implies that $s \vdash [\alpha^*]\varphi$ iff, for all i, $s \vdash [\alpha^i]\varphi$. Then definitions (2), (8), and (10) of the inductive case imply that $s \vdash (\theta \vee [\alpha^*]\varphi)$ iff for all i, $s \vdash (\theta \vee [\alpha^i]\varphi)$. So in Kripke models an infinite distributive law for θ over $[\alpha^*]$, with terms $[\alpha^i]$, is valid, a special case of a distributive law not valid in all complete Heyting algebras. On the other hand, in Kripke models, $s \vdash < \alpha^* > \varphi$ iff there is an i such that $s \vdash < \alpha^i > \varphi$. This is a special case of an infinite distributive law holding outright in all complete Heyting valuations of ordinary intuitionistic logic.

Given a possible world $k \in K$ and a partial state $s : T \rightarrow D(k)$, we have now defined two notions of forcing. One for forcing by the possible world k for formulas of first order logic. The other is the forcing by the state s for formulas of dynamic logic. Here we show the connection between the two notions for $\diamond$, $\Box$-free formulas. It can be shown by induction on the length of the formula that such formulas are built up from $\wedge$, $\vee$, $\rightarrow$, $\neg$, $\forall$, and $\exists$. We note that if we used only a single kind of variable then we could have defined both notions of forcing simultaneously.

Lemma 4.2.8. *Suppose that $k \in K$ is a possible world and $s : T \rightarrow D(k)$ is a partial state. Suppose that φ is a formula in CCDL that does not have either $\diamond$ or $\Box$, or any unquantified logic variables. Let $t_1, \ldots, t_n$ be the list of all program terms that appear in φ*

[5] $s' : X \rightarrow D(k')$.

[6] Note. Schemas 7 and 8 are inductive clauses defining forcing from both accessibility and forcing.

and are defined in s. Let $s(t_1), \ldots, s(t_n)$ *be their denotations in* $D(k)$. *Then* $s \vdash \varphi$ *if and only if* $k \vdash \varphi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$.

Proof. By induction on the structure of φ.
Case 1. φ is the atomic formula $A_i(t_1, \ldots, t_n)$.
Then by Definition 4.2.7 and 4.2.3, $s \vdash A_i(t_1, \ldots, t_n)$ if and only if $A_{i,k}(s(t_1), \ldots, s(t_n))$. This is satisfied if and only if $k \vdash \varphi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$.
Case 2. φ is $t_i = t_j$.
Then $s \vdash (t_i = t_j)$ if and only if $s(t_i) =_k s(t_j)$ if and only if $k \vdash (s(t_i) = s(t_j))$.
Cases 3 and 4. φ is $\varphi_1 \wedge \varphi_2$, or $\varphi i \varphi_1 \vee \varphi_2$.
These are straightforward applications of Definition 4.2.7.
Case 5. φ is $\forall y \psi(y)$.
Suppose that $s \vdash \varphi$. To show that $k \vdash \varphi$ take any $k' \geq k$. Then there is a partial state $s' \geq s$ satisfying $s' : T \to D(k')$ and $s' \vdash \psi(d)$ for any $d \in D(k')$. Hence by the inductive assumption $k' \vdash \psi(d, s'(t_1)/t_1, \ldots, s'(t_n)/t_n)$. Therefore $k \vdash (\forall y \psi(y))$. Conversely suppose that $k \vdash (\forall y \psi(y))(s(t_1)/t_1, \ldots, s(t_n)/t_n)$. To show that $s \vdash (\forall y \psi(y))$ take any $s' \geq s$. Then there is $k' \geq k$ with $s' : T \to D(k')$. Now pick any $d \in D(k')$. Then $k' \vdash \psi(d/y, s(t_1)/t_1, \ldots, s(t_n)/t_n)$. Now, by the inductive assumption $s' \vdash \psi(d/y)$. Hence $s \vdash \forall y \psi(y)$.
Case 6. φ is $\exists y \psi(y)$.
Suppose that $s \vdash \exists y \psi(y)$. Then there is a $d \in D(k)$ with $s \vdash \psi(d)$. Now by the inductive assumption $k \vdash \psi(d/y, s(t_1)/t_1, \ldots, s(t_n)/t_n)$. Hence $k \vdash \exists y \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$. This argument is reversible.
Case 7. φ is $\psi \to \theta$.
Suppose that $s \vdash \varphi$; i.e. $s \vdash \psi \to \theta$. To show that $k \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n) \to \theta(s(t_1)/t_1, \ldots, s(t_n)/t_n)$ take any $k' \geq k$. Then there is $s' \geq s$ with $s' : T \to D(k')$. Now suppose that $k' \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$. Then, by the inductive assumption $s' \vdash \psi$. Hence $s' \vdash \theta$. Therefore $k' \vdash \theta(s(t_1)/t_1, \ldots, s(t_n)/t_n)$ once again by the inductive assumption. Hence $k \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n) \to \theta(s(t_1)/t_1, \ldots, s(t_n)/t_n)$.

Now conversely suppose that $k \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n \to \theta(s(t_1)/t_1, \ldots, s(t_n)/t_n)$. To show that $s \vdash \psi \to \theta$, take any $s' \geq s$ with $s' \vdash \psi$. Then there is $k' \geq k$ with $s' : T \to k'$. The inductive assumption implies that $k' \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$, therefore giving that $k' \vdash \theta(s(t_1)/t_1, \ldots, s(t_n)/t_n)$. Now the inductive assumption gives that $s' \vdash \theta$, giving us $s \vdash \psi \to \theta$.
Case 8. φ is $\neg\psi$.
Suppose that $s \vdash \neg\psi$. To show that $k \vdash \neg\psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$, assume that there is $k' \geq k$ with $k' \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$. Then there is $s' \geq s$ with $s' : T \to D(k')$. Now applying the inductive assumption to s' we get that $s' \vdash \psi$, contradicting the assumption $s \vdash \neg\psi$.

Conversely if $k \vdash \neg\psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$, to show that $s \vdash \neg\psi$, assume for a contradiction that there is $s' \geq s$ satisfying $s' \vdash \psi$. Then there is $k' \geq k$ satisfying $s' : T \to D(k')$. Applying the inductive assumption to s' will give that $k' \vdash \psi(s(t_1)/t_1, \ldots, s(t_n)/t_n)$ for a contradiction. □

We use the next lemma to show the monotonicity of forcing.

Lemma 4.2.9. *Let α be any program, $(s, S') \in R$ and $s' \geq s$ be partial states. Then there is $S'' \subseteq S$ with $(s', S'') \in R(\alpha)$ satisfying the condition that for each $s'' \in S''$ there is a $u \in S'$ satisfying $u \leq s''$.*

Proof. By induction on the structure of α.
Case 1. α is $x_i \leftarrow t$.
Then $S' = \{s(s(t)/x_i)\}$. Now take $S'' = \{s'(s'(t)/x_i)\}$. Notice that $(s', S'') \in R(\alpha)$ and $s'(s'(t)/x_i) \geq s(s(t)/x_i)$.
Case 2. α is φ? for $\Diamond, \Box$-free φ.
Suppose that $(s, S') \in R(\alpha)$. Then $S' = \{s\}$ and $s \vdash \varphi$. If $s' \geq s$ then by Lemma 4.2.7 we get that $k \vdash \varphi$. By monotonicity of forcing for ordinary intuitionistic logic we get $k' \vdash \varphi$ where $s' \to D(k')$. Then by Lemma 4.2.7 we get $s' \vdash \varphi$. Hence $(s', \{s'\}) \in R(\alpha)$.
Case 3. Suppose that α is $(\beta; \gamma)$.
Suppose that $(s, S') \in R(\alpha)$ and $s' \geq s$. Definition of $(\beta; \gamma)$ provides us with $U \subseteq S$ with $(s, U) \in R(\beta)$ and for each $u \in U$ an $S'_u \subseteq S'$ is satisfying $(u, S'_u) \in R(\gamma)$ and $S' =_u \cup\{S'_u : u \in U\}$.

The inductive assumption applied to $s' \geq s$ and $(s, U) \in R(\beta)$ gives that there is $V \subseteq S$ with $(s', V) \in R(\beta)$ satisfying the condition that for each $v \in V$ there is $u \in U$ satisfying $u \leq v$. The inductive assumption applied to each $v \in V$ and the corresponding $u \in U$ with $u \leq v$ and $(u, S'_u) \in R(\gamma)$ we get W_v with $(v, W_v) \in R(\gamma)$ and for each $w \in W_v$, there exists $q \in S'_u$ with $w \geq q$. Then notice that $(s', \cup_{v \in V} W_v) \in R(\gamma)$ and for each $w \in \cup_{v \in V} W_v$ there is $q \in S'$ satisfying $q \leq w$.
Case 4. α is $\beta \cup \gamma$.
Suppose that $(s, S') \in R(\alpha)$ and $s' \geq s$. Then $(s, S') \in R(\beta)$ or $(s, S') \in R(\gamma)$. If $(s, S') \in R(\beta)$, the inductive assumption gives $V \subseteq S$ with $(s', V) \in R(\beta)$ and for each $v \in V$ there is $w \in S'$ with $w \leq v$. The same holds if $(s, S') \in R(\gamma)$.
Case 5. α is $\beta \cap \gamma$.
Then there exists S'_1, S'_2 with $(s, S'_1) \in R(\beta)$, $(s, S'_2) \in R(\gamma)$ and $S'_1 \cup S'_2 = S'$. Now by the inductive assumption there is $V_1, V_2 \subseteq S$ satisfying $(s', V_1) \in R(\beta)$, $(\beta', V_2) \in R(\gamma)$, and for each $v_1 \in V_1$ there is $w_1 \in S'_1$ with $v_1 \geq w_1$ and for each $v_2 \in V_2$ there exists $w_2 \in S_2$ with $v_2 \geq w_2$. Then $(s', V \cup V) \in R(\beta \cap \gamma)$ with the required properties. □

We use this lemma to prove monotonicity of forcing.

Lemma 4.2.10 (*Monotonicity of Forcing*). *Suppose that K is a Kripke frame and s, s' are states over K that satisfy $s \leq s'$. Then $s \vdash \varphi$ implies $s' \vdash \varphi$ for any formula φ of CCDL.*

Proof. By induction on the definition of forcing. The inductive case of $< \alpha > \varphi$ uses the previous lemma. □

Definition 4.2.11 (*Substitution*). Let $\bar{t}$ be a vector of program terms. Let $\bar{x}$ be a vector of program variables. Let φ be a sentence. Define $\varphi(\bar{t}/\bar{x})$, the result of substituting $\bar{t}$ for $\bar{x}$ in φ, by induction on the definition of formula as given below.

- If φ is $A(\bar{x})$, then $\varphi(\bar{t}/\bar{x})$ is $A(\bar{t}/\bar{x})$.
- If φ is $t_1 = t_2$, then $\varphi(\bar{t}/\bar{x})$ is $t_1(\bar{t}/\bar{x}) = t_2(\bar{t}/\bar{x})$.
- If φ is $\psi \wedge \gamma$, $\psi \vee \gamma$, $\psi \to \gamma$, $\forall y \psi$, $\exists y \psi$, then $\varphi(\bar{t}/\bar{x})$ is, respectively, $\psi(\bar{t}/\bar{x}) \wedge \gamma(\bar{t}/\bar{x})$, $\psi(\bar{t}/\bar{x}) \vee \gamma(\bar{t}/\bar{x})$, $\psi(\bar{t}/\bar{x}) \to \gamma(\bar{t}/\bar{x})$, $\forall y \psi(\bar{t}/\bar{x})$, $\exists y \psi(\bar{t}/\bar{x})$.

- If φ is either $< \bar{z} \leftarrow \bar{x}; \alpha; \bar{x} \leftarrow \bar{z} > \varphi$, or $[\bar{z} \leftarrow \bar{x}; \alpha; \bar{x} \leftarrow \bar{z}]\varphi$[7], where $\bar{x}$ does not appear in α and $\bar{z}$ does not appear in φ, then $\varphi(\bar{t}/\bar{x})$ is respectively $< \bar{z} \leftarrow \bar{t}; \alpha; \bar{x} \leftarrow \bar{z} > \varphi$ or $[\bar{z} \leftarrow \bar{t}; \alpha; \bar{x} \leftarrow \bar{z}]\varphi$.
- If φ is $< \alpha > \psi$ or $[\alpha]\psi$ then define $\varphi(\bar{t}/\bar{x})$ respectively as $(< \bar{z} \leftarrow \bar{x}; \alpha(\bar{z}/\bar{x}); \bar{x} \leftarrow \bar{z} > \psi)(\bar{t}/\bar{x})$ or $([\bar{z} \leftarrow \bar{x}; \alpha(\bar{z}/\bar{x}); \bar{x} \leftarrow \bar{z}]\psi)(\bar{t}/\bar{x})$ with $\bar{z}$ a new vector of program variables not occurring in α. Here $\alpha(\bar{z}/\bar{x})$ is literal substitution. □

5. Axioms of CCDL and some consequences

We give some non-logical axioms of constructive concurrent dynamic logic (CCDL). We use them to derive some useful equivalences using intuitionistic logic with modal operators allowed in the formulas. The semantical correctness of these axioms for the intended models of CCDL will be verified at length in a later section. These axioms are rewrite rules which simplify certain CCDL formulas. They are later incorporated into the definition of CCDL tableau proofs. They will also be used in the projected sequel on finitary CCDL as term extraction algorithms.

5.1. Axioms of CCDL

1 Axioms of modal logic
1.1 Axioms for the box operator

Associativity of composition

B1.1A $[\alpha_1; (\alpha_2; \alpha_3)]\varphi \leftrightarrow [(\alpha_1; \alpha_2); \alpha_3]\varphi$.
B1.1B $[(\alpha_1; (\alpha_2; \alpha_3)) \cap \beta]\varphi \leftrightarrow [((\alpha_1; \alpha_2); \alpha_3) \cap \beta]\varphi$.
B1.2A $[\alpha_0; ((\alpha_1; \alpha_2); \alpha_3)]\varphi \leftrightarrow [\alpha_0; (\alpha_1; (\alpha_2; \alpha_3))]\varphi$.
B1.2B $[(\alpha_0; ((\alpha_1; \alpha_2); \alpha_3)) \cap \beta]\varphi \leftrightarrow [(\alpha_0; (\alpha_1; (\alpha_2; \alpha_3))) \cap \beta]\varphi$.

Associativity and commutativity of the parallel $\cap$ operator

B2.1A $[(\alpha \cap \beta) \cap \gamma]\varphi \leftrightarrow [\alpha \cap (\beta \cap \gamma)]\varphi$.
B2.1B $[((\alpha \cap \beta) \cap \gamma) \cap \delta]\varphi \leftrightarrow [(\alpha \cap (\beta \cap \gamma)) \cap \delta]\varphi$.
B2.2 $[\alpha \cap \beta]\varphi \leftrightarrow [\beta \cap \alpha]\varphi$.
B2.3A $[(\alpha \cap \beta); \gamma]\varphi \leftrightarrow [(\alpha; \gamma) \cap (\beta; \gamma)]\varphi$.
B2.3B $[((\alpha \cap \beta); \gamma) \cap \delta]\varphi \leftrightarrow [(\alpha; \gamma) \cap (\beta; \gamma) \cap \delta]\varphi$.
B2.4A For α free of $\cup, \cap$ and $*$
$[\alpha; (\beta \cap \gamma)]\varphi \leftrightarrow [(\alpha; \beta) \cap (\alpha; \gamma)]\varphi$.
B2.4B For α free of $\cup, \cap$ and $*$
$[(\alpha; (\beta \cap \gamma)) \cap \delta]\varphi \leftrightarrow [((\alpha; \beta) \cap (\alpha; \gamma)) \cap \delta]\varphi$.

Associativity and commutativity of non-deterministic choice $\cup$

B3.1 $[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$.
B3.2 $[(\alpha \cup \beta); \theta]\varphi \leftrightarrow [(\alpha; \theta) \cup (\beta; \theta)]\varphi$.

[7] Convention: If $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{z} = (z_1, \ldots, z_n)$, then we let $\bar{x} \leftarrow \bar{z}$ abbreviate $(x_1 \leftarrow z_1; \ldots; x_n \leftarrow z_n)$.

B3.3 For α free of $\cap, \cup$ and $*$
$[\alpha; (\beta \cup \gamma)]\varphi \leftrightarrow [(\alpha; \beta) \cup (\alpha; \gamma)]\varphi$.
B3.4 For α free of $\cap, \cup$ and $*$
$[(\alpha; (\beta \cup \gamma)) \cap \delta]\varphi \leftrightarrow [((\alpha; \beta) \cup (\alpha; \gamma)) \cap \delta]\varphi$.

Idempotency of $\cap$ inside the box operator
B4.1A $[\alpha \cap \alpha]\varphi \leftrightarrow [\alpha]\varphi$.
B4.1B $[\alpha \cap \alpha \cap \beta]\varphi \leftrightarrow [\alpha \cap \beta]\varphi$.
B4.1C $[(\alpha \cap \alpha); \theta]\varphi \leftrightarrow [\alpha; \theta]\varphi$.
B4.1D $[(\alpha \cap \alpha \cap \beta); \theta]\varphi \leftrightarrow [(\alpha \cap \beta); \theta]\varphi$.

Distributivity of $\cap$ and $\cup$
B5.1 $[(\alpha \cup \beta) \cap \delta]\varphi \leftrightarrow [(\alpha \cap \delta) \cup (\beta \cap \delta)]\varphi$.

Test operator ?
B6.1 $[\theta?]\varphi \leftrightarrow (\theta \rightarrow \varphi)$.

1.2 Axioms for the diamond operator

Associativity of composition
D1.1 $< \alpha; \beta > \varphi \leftrightarrow < \alpha > (< \beta > \varphi)$.
D1.2 $< (\alpha; (\beta; \gamma)) > \varphi \leftrightarrow < ((\alpha; \beta); \gamma)) > \varphi$.

Associativity and commutativity of $\cap$
D2.1 $< \alpha \cap \beta > \varphi \leftrightarrow < \alpha > \varphi \wedge < \beta > \varphi$.
D2.2 For α free of $\cap, \cup$ and $*$
$< \alpha; (\beta \cap \gamma) > \varphi \leftrightarrow < (\alpha; \beta) \cap (\alpha; \gamma) > \varphi$.

Associativity and commutativity of $\cup$
D3.1 $< \alpha \cup \beta > \varphi \leftrightarrow < \alpha > \varphi \vee < \beta > \varphi$.
D3.2 For α free of $\cup, \cap$ and $*$
$< \alpha; (\beta \cup \gamma) > \varphi \leftrightarrow < (\alpha; \beta) \cup (\alpha; \gamma) > \varphi$.

Test operator ?
D4 $< \theta? > \varphi \leftrightarrow \theta \wedge \varphi$.

2 Axioms translating modal operators into first order logic

D-trans $< x_i \leftarrow t > \varphi \leftrightarrow \exists y(y = t \wedge \varphi(t/x_i))$ where t is a program term.
B-trans $[(\alpha_{11}; \alpha_{12}; \ldots; \alpha_{1n_1}) \cap \cdots \cap (\alpha_{k1}; \alpha_{k2}; \ldots; a_{kn_k})]\varphi \leftrightarrow \psi$ where each α_{ij} is either an assignment or a first order test and $\bar{x}$ is the n-ary vector of all program variables appearing in the program, and ψ is defined as follows.

- First, let $\{\bar{r}_{ij} : i \leq k, j \leq n_k\}$ be a double sequence of n-ary program variables defined as follows.

1. Let $\bar{r}_{i,0}$ be $\bar{x}$.
2. If α_{ij} is $x_m \leftarrow f(\bar{x})$ let $\bar{r}_{i,j}$ be $\bar{r}_{i,j-1}(f(\bar{r}_{i,j-1})/\bar{r}_{i,j,m})$ where $\bar{r}_{i,j,m}$ is the mth position of $\bar{r}_{i,j-1}$.
3. If not let $\bar{r}_{ij}$ be $\bar{r}_{i,j-1}$.

- Define λ_i as $\wedge\{\alpha_{i,j}(\bar{r}_{i,j}) : \alpha_{i,j}$ is a test$\}$.
- Define λ as $\wedge_i \lambda_i$.
- Define η as $\wedge_{i,j}(\exists \bar{y}_{ij})(\bar{y}_{ij} = \bar{r}_{ij})$.
- Then ψ is $(\eta \wedge \lambda) \rightarrow \wedge\{\varphi(\bar{r}_{i,n_i}/\bar{x}) : i \leq k\}$.

3 Equality axioms

Pure equality

eq1 $\forall y(y = y)$.

eq2 $\forall y_1 y_2(y_1 = y_2 \rightarrow y_2 = y_1)$.

eq3 $\forall y_1 y_2 y_3((y_1 = y_2) \wedge (y_2 = y_3) \rightarrow (y_1 = y_3))$.

Predicate symbols

Pred $\forall y_1, \ldots, y_m, y'_1, \ldots, y'_n(y_1 = y'_1 \wedge \cdots \wedge y_n = y'_n \wedge A(y_1, \ldots, y_n) \rightarrow A(y'_1, \ldots, y'_n))$ where n is the arity of the predicate symbol.

Function symbols

fun1 $\forall y_1, \ldots, y_n y(y = g(y_1, \ldots, y_n) \rightarrow \exists y'_1, \ldots, y'_n(y_1 = y'_1 \wedge \cdots \wedge y_n = y'_n))$.

fun2 $\forall y_1, \ldots, y_n, y'_1, \ldots, y'_n(y_1 = y'_1, \ldots, y_n = y'_n \rightarrow g(y_1, \ldots, y_n) = g(y'_1, \ldots, y'_n))$.

fun3 $\forall y_1, \ldots, y_n, y, y_0(y = g(y_1, \ldots, y_n) \wedge y_0 = g(y_1, \ldots, y_n) \rightarrow y = y_0)$ where n is the arity of the function symbol g.

Existence of state variables

Sv $\exists y(y = x_i)$.

5.2. Some consequences

The following lemma and its corollaries are here to show the associativity of the composition operator, in programs that are solely constructed by compositions.

Lemma 5.2.1. *Let $\beta, \alpha_1, \alpha_2, \ldots, \alpha_n$ be programs. Let α be formed by compositions of $\alpha_1, \alpha_2, \ldots, \alpha_n$ and let them appear from left to right in that order (i.e. with any arbitrary associativity of ;). Then*

$$[\beta; \alpha]\varphi \leftrightarrow [\beta; (\alpha_1; (\alpha_2; (\alpha_3; \ldots; (\alpha_{n-1}; \alpha_n)]\varphi.$$

Proof. By induction on n.
Case $n = 1$ is trivial.
Suppose that $n > 1$. Let $\alpha = \theta; \gamma$ where θ has $\alpha_1, \ldots, \alpha_k$ and γ has $\alpha_{k+1}, \ldots, \alpha_n$ for some k with $1 < k < n$. Then

$$[\beta; \alpha]\varphi \leftrightarrow [\beta; (\theta; \gamma)]\varphi \leftrightarrow [(\beta; \theta); \gamma]\varphi$$

by axiom B1.1A. Now if $k + 1 < n$ then by the inductive assumption for $k + 1$

$$[(\beta; \theta); \gamma]\varphi \leftrightarrow [(\beta; \theta); (\alpha_{k+1}; (\alpha_{k+2}; (\ldots; (\alpha_{n-1}; \alpha_n)]\varphi$$

and by the induction assumption for $n = 2$ we get

$$\leftrightarrow [\beta; (\theta; (\alpha_{k+1}; (\alpha_{k+2}; \ldots (\alpha_{n-1}; \alpha_n)]\varphi.$$

By the inductive assumption for $k + 1$ we get

$$\leftrightarrow [\beta; (\alpha_1; (\alpha_2; \ldots, (\alpha_{n-1}; \alpha_n))]\varphi.$$

If $k + 1 = n$ and $n = 2$ then its axiom B1.1A; else if $k + 1 = n$ and $n > 2$, let $\theta = \theta_1; \theta_2$; then

$$[\beta; (\theta; \alpha_n)]\varphi \leftrightarrow [\beta; (\theta_1; (\theta_2; \alpha_n))]\varphi \text{ by axiom B1.1A}$$
$$\leftrightarrow [(\beta; \theta_1); (\theta_2; \alpha_n)]\varphi \text{ again by axiom B1.1A.}$$

Now applying the inductive assumption to $(\theta_2; \alpha_n)$ (if θ_2 has α_ℓ to α_{n-1}) we get

$$\leftrightarrow [(\beta; \theta_1); (\alpha_\ell; (\alpha_{\ell+1}, \ldots, (\alpha_{n-1}; \alpha_n))]\varphi \text{ by axiom B1.1A}$$
$$\leftrightarrow [\beta; (\theta_1; (\alpha_\ell; (\alpha_{\ell+1}; \ldots (\alpha_{n-1}; \alpha_n)))]\varphi.$$

Now apply the inductive assumption to $[\beta; (\theta_1; \lambda)]\varphi$ where

$$\lambda = \alpha_\ell; (\alpha_{\ell+1}; (\alpha \ldots (\alpha_{n-1}; \alpha_n).$$

Then we get

$$\leftrightarrow [\beta; (\alpha_1; \alpha_2; (\alpha_3; (\ldots; (\alpha_{n-1}; \alpha_n))]\varphi. \quad \square$$

Corollary 5.2.2. *If α is as given in the statement of Lemma 5.2.1, then*

$$[\alpha]\varphi \leftrightarrow [\alpha_1; (\alpha_2; \ldots; (\alpha_{n-1}; \alpha_n))]\varphi.$$

Proof. By induction on n.
For $n > 1$ let α be $\beta; \gamma$ where β consists of $\alpha_1, \ldots, \alpha_{\ell-1}$ and γ consists of $\alpha_\ell, \ldots, \alpha_n$; then

$$[\alpha]\varphi \leftrightarrow [\beta; \gamma]\varphi \leftrightarrow [\beta; \gamma']\varphi$$

by Lemma 5.2.1 where

$$\gamma' = \alpha_\ell; (\alpha_{\ell+1}; \ldots; (\alpha_{n-1}; \alpha_n))).$$

Now applying the inductive assumption to $[\beta; \gamma']\varphi$ will give the desired result. $\square$

Lemma 5.2.3. *Let $\beta, \alpha_1, \ldots, \alpha_n, \delta$ be programs. Let α be formed by composition of $\alpha_1, \alpha_2, \ldots, \alpha_n$ and let them appear from left to right in that order (i.e. with any arbitrary associativity of ;); then*

$$[(\beta; \alpha) \cap \delta]\varphi \leftrightarrow [(\beta; (\alpha_1; (\alpha_2; (\ldots (\alpha_{n-1}; \alpha_n))) \cap \delta]\varphi.$$

Proof. This is similar to the proof of Lemma 5.2.1. $\square$

Corollary 5.2.4. *If α satisfy the hypothesis of Lemma 5.2.3 and δ is any program, then*

$$[\alpha \cap \delta]\varphi \leftrightarrow [(\alpha_1; (\alpha_2; \ldots (\alpha_{n-1}; \alpha_n))) \cap \delta]\varphi.$$

Proof. Similar to the proof of Corollary 5.2.2. □

Corollary 5.2.5. *If α satisfies the hypothesis of Lemma 5.2.1, then*

$$[\alpha]\varphi \leftrightarrow [((\alpha_1; \alpha_2); \alpha_3); \ldots; \alpha_n)]\varphi.$$

Proof. Apply Corollary 5.2.2 to

$$[((\alpha_1; \alpha_2); \alpha_3); \alpha_4 \ldots; \alpha_n)]\varphi$$

and then we get

$$[((\alpha_1; \alpha_2); \alpha_3; \ldots; \alpha_n]\varphi \leftrightarrow [\alpha_1; (\alpha_2; (\ldots (\alpha_{n-1}; \alpha_n))]\varphi.$$

Now by Corollary 5.2.2 we get $[\alpha]\varphi$ equivalent to the right hand side. □

Corollary 5.2.6. *If α satisfies the hypothesis of Lemma 5.2.1, then*

$$[\alpha \cap \delta]\varphi \leftrightarrow [((\alpha_1; \alpha_2); \alpha_3) \ldots; \alpha_n) \cap \delta]\varphi.$$

Proof. Similar to the proof of Corollary 5.2.5 (use Lemma 5.2.3 instead of Corollary 5.2.2). □

For the associativity of the $\cap$ operator, we prove the following lemma.

Lemma 5.2.7. *Suppose that α is constructed out of $\alpha_1, \ldots, \alpha_n$, and the $\cap$ operators and in α from left to right $\alpha_1, \ldots, \alpha_n$ appear in that order, then the following are true.*

$$[\alpha]\varphi \leftrightarrow [\alpha_1 \cap (\alpha_2 \cap (\alpha_3 \cap (\ldots \cap (\alpha_{n-1} \cap \alpha_n))))]\varphi$$
$$[\alpha \cap \beta]\varphi \leftrightarrow [(\alpha_1 \cap (\alpha_2 \cap \cdots \cap (\alpha_{n-1} \cap \alpha_n))) \cap \beta]\varphi$$
$$[\alpha; \theta]\varphi \leftrightarrow [(\alpha_1 \cap (\alpha_2 \cap \cdots \cap (\alpha_{n-1} \cap \alpha_n))); \theta]\varphi.$$

Proof. The proofs are by induction on n and have the same flavour as the corollaries and Lemma 5.2.1 through 5.2.6. □

Notice that axiom B3.1 implies that the unions in $[\alpha_1 \cup \alpha_2 \cup \cdots \cup \alpha_n]\varphi$ are both commutative and associative. Also Lemma 5.2.7 implies that the intersections in $[\alpha_1 \cap \alpha_2 \cap \cdots \cap \alpha_n]\varphi$ are both commutative and associative. Likewise the compositions in $[\alpha_1; \alpha_2; \ldots; \alpha_n]\varphi$ are associative by Lemma 5.2.1. For the sake of readability hereafter we omit extra parentheses used to indicate explicit associativity of $\cup, \cap$ and ; in these expressions.

6. Normal forms of programs

In this section we define the concept of a *normal form* of a program. We show that any program α has a normal form (say α') having the properties that for any proposition φ,

- $< \alpha > \varphi \leftrightarrow < \alpha' > \varphi$,
- $[\alpha]\varphi \leftrightarrow [\alpha']\varphi$.

When the program α has no $*$ in it, its normal form will reduce to what we call the *finite normal form*. We use this name because a formula that has only programs in finite normal form is equivalent to a first order formula with no modal operators. This fact is proved later.

6.1. Some preliminary results

Definition 6.1.1 (*$*$-free Program in Finite Normal Form*). A $*$-free program α is said to be in finite normal form if α is of the form

- $\alpha_1 \cup \alpha_2 \cup \cdots \cup \alpha_n$ for some n where each α_i is of the form
- $\alpha_{i,1} \cap \alpha_{i,2} \cap \cdots \cap \alpha_{i,n_i}$ where each $\alpha_{i,k}$ is of the form
- $\theta_{i,k,1}; \theta_{i,k,2}; \ldots; \theta_{i,k,m_{i,k}}$ where each $\theta_{i,k,j}$ is
- either an atomic assignment or a first order test.

In order to prove the normal form theorem (i.e. that every $*$-free program has a finite normal form) we need some preliminary results such as $\cup$ and $\cap$ appearing inside a box satisfying commutative and associative laws and composition (;) appearing inside a box being associative.

Lemmas 6.1.2 through 6.1.9 show that for programs α, β in finite normal form the $\cup$ and $\cap$ are both commutative and associative and composition is associative in $[\alpha \cap \beta]\varphi$.

Lemma 6.1.2. *For any programs α, β, γ and any proposition φ,*

1. $[(\alpha \cup \beta) \cap \delta]\varphi \leftrightarrow [(\beta \cup \alpha) \cap \delta]\varphi$,
2. $[\delta \cap (\alpha \cup \beta)]\varphi \leftrightarrow [\delta \cap (\beta \cup \alpha)]\varphi$.

Proof. (2) follows from (1) by axiom B2.2A.
To prove (1);

$$
\begin{aligned}
[(\alpha \cup \beta) \cap \delta]\varphi &\leftrightarrow [(\alpha \cap \delta) \cup (\beta \cap \delta)]\varphi && (1)\\
&\leftrightarrow [\alpha \cap \delta]\varphi \wedge [\beta \cap \delta]\varphi && (2)\\
&\leftrightarrow [\beta \cap \delta]\varphi \wedge [\alpha \cap \delta]\varphi && (3)\\
&\leftrightarrow [(\beta \cup \alpha) \cap \delta]\varphi. && (4)
\end{aligned}
$$

The justifications are: (1) by axiom B5.1, (2) by axiom B3.1, (3) by commutativity of $\wedge$, and (4) by axioms B3.1 and B5.1. $\square$

Lemma 6.1.3. *For any programs $\alpha, \beta, \gamma, \delta$ and any proposition φ,*

1. $[((\alpha \cup \beta) \cup \gamma) \cap \delta]\varphi \leftrightarrow [(\alpha \cup (\beta \cup \gamma)) \cap \delta]\varphi$,
2. $[\delta \cap ((\alpha \cup \beta) \cup \gamma)]\varphi \leftrightarrow [\delta \cap ((\alpha \cup (\beta \cup \gamma))]\varphi$.

Proof. By using axioms B5.1 and B3.1 as in Lemma 6.1.2 we can show that both the left and right hand sides are equivalent to

$$[\alpha \cap \delta]\varphi \wedge [\beta \cap \delta]\varphi \wedge [\gamma \cap \delta]\varphi. \quad \square$$

Lemma 6.1.4. *For any programs $\alpha_1, \alpha_2, \beta$, and δ and any proposition φ,*

1. $[((\alpha_1 \cap \alpha_2) \cup \beta) \cap \delta]\varphi \leftrightarrow [((\alpha_2 \cap \alpha_1) \cup \beta) \cap \delta]\varphi$,
2. $[\delta \cap ((\alpha_1 \cap \alpha_2) \cup \beta)]\varphi \leftrightarrow [\delta \cap ((\alpha_2 \cap \alpha_1) \cup \beta)]\varphi$.

Proof. By using axioms B2.2, B5.1, and B3.1 we can show that both sides are equivalent to

$$[\alpha_1 \cap \alpha_2 \cap \delta]\varphi \wedge [\beta \cap \delta]\varphi. \quad \square$$

Lemma 6.1.5. *For any programs $\alpha_1, \alpha_2, \alpha_3, \beta, \delta$ and any proposition φ,*

1. $[(((\alpha_1 \cap \alpha_2) \cap \alpha_3) \cup \beta) \cap \delta]\varphi \leftrightarrow [(((\alpha_1 \cap (\alpha_2 \cap \alpha_3))) \cup \beta) \cap \delta]\varphi$,
2. $[\delta \cap ((\alpha_1 \cap \alpha_2) \cap \alpha_3)]\varphi \leftrightarrow [\delta \cap (\alpha_1 \cap (\alpha_2 \cap \alpha_3))]\varphi$.

Proof. For the proof of (1), it can be shown that both sides are equivalent to

$$[\alpha_1 \cap \alpha_2 \cap \alpha_3 \cap \delta]\varphi \wedge [\beta \cap \delta]\varphi$$

by using axioms B5.1, B3.1, and B2.1. Proof of (2) is a special case of the proof of (1). $\square$

Lemma 6.1.6. *For any programs $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta$, and δ,*

1. $[((((\alpha_1; \alpha_2); \alpha_3) \cap \alpha_4) \cup \beta) \cap \delta]\varphi \leftrightarrow [(((\alpha_1; (\alpha_2; \alpha_3)) \cap \alpha_4) \cup \beta) \cap \delta]\varphi$,
2. $[\delta \cap (((\alpha_1; \alpha_2); \alpha_3) \cap \alpha_4)]\varphi \leftrightarrow [\delta \cap ((\alpha_1; (\alpha_2; \alpha_3)) \cap \alpha_4)]\varphi$.

Proof. We can show that both sides are equivalent to

$$[(\alpha_1; \alpha_2; \alpha_3) \cap \alpha_4 \cap \delta]\varphi \wedge [\beta \cap \delta]\varphi$$

by using axioms B1.1B, B5.1, and B3.1. $\square$

Notice that Lemmas 6.1.2 through 6.1.6 imply that for programs α, β appearing in finite normal form and for any proposition φ, the $\cap$'s and $\cup$'s appearing in $[\alpha \cap \beta]\varphi$ are both commutative and associative and also that compositions are associative. The Lemmas 6.1.7 through 6.1.9 imply the associativity of the composition operator (;).

Lemma 6.1.7. *For any programs $\alpha, \alpha_i, \theta, \beta, \gamma, \delta$ and any proposition φ,*

1. $[(\alpha_1 \cup \alpha_2); \theta]\varphi \leftrightarrow [(\alpha_2 \cup \alpha_1); \theta]\varphi$,
2. $[((\alpha \cup \beta) \cup \gamma); \theta]\varphi \leftrightarrow [(\alpha \cup (\beta \cup \gamma)); \theta]\varphi$,
3. $[(\alpha \cap \beta); \gamma]\varphi \leftrightarrow [(\beta \cap \alpha); \gamma]\varphi$,
4. $[(\alpha \cap (\beta \cap \delta)); \gamma]\varphi \leftrightarrow [((\alpha \cap \beta) \cap \delta); \gamma]\varphi$.

Proof.

1. Axioms B3.1 and B3.2 imply that both sides are equivalent to $[\alpha_1; \theta]\varphi \wedge [\alpha_2; \theta]\varphi$.
2. Axioms B3.1 and B3.2 imply that both sides are equivalent to $[\alpha; \theta]\varphi \wedge [\beta; \theta]\varphi \wedge [\gamma; \theta]\varphi$.
3. Axioms B2.3A and B2.2 imply that both sides are equivalent to $[(\alpha; \gamma) \cap (\beta; \gamma)]\varphi$.
4. Axioms B2.3B, B2.3A, and B2.2 imply that both sides are equivalent to $[(\alpha; \gamma) \cap (\beta; \gamma) \cap (\delta; \gamma)]\varphi$. $\square$

Notice that by Lemma 6.1.7 the order of associativity of $\cap, \cup$ over the α_i's does not matter in $[(\cap_{i=1}^{n} \alpha_i); \theta]\varphi$ and $[(\cup_{i=1}^{n} \alpha_i); \theta]\varphi$.

Lemma 6.1.8. *For any programs α_i, θ and any proposition φ,*

1.
$$\left[\left(\bigcup_{i=1}^{n}\alpha_i\right);\theta\right]\varphi \leftrightarrow \bigwedge_{i=1}^{n}[\alpha_i;\theta]\varphi,$$
2.
$$\left[\left(\bigcap_{i=1}^{n}\alpha_i\right);\theta\right]\varphi \leftrightarrow \left[\bigcap_{i=1}^{n}(\alpha_i;\theta)\right]\varphi.$$

Proof. (1) is proved by repeated application of axioms B3.1 and B3.2, and (2) is proved by repeated application of axioms B2.3A, B2.3B, and B2.2. □

Lemma 6.1.9. *Suppose that α in finite normal form is $\cup_{i=1}^{n}\alpha_i$ where each α_i is $\cap_{j=1}^{n_i}\alpha_{ij}$, where each α_{ij} is a composition of a finite number of tests and assignments. Then*

$$[\alpha;\theta]\varphi \leftrightarrow \bigwedge_{i=1}^{n}\left[\bigcap_{j=1}^{n_i}(\alpha_{i,j};\theta)\right]\varphi.$$

Therefore the compositions that occur in α_{ij} are associative and the $\cup$'s and $\cap$'s that appear in α are both commutative and associative.

Proof.

$$[\alpha;\theta]\varphi \leftrightarrow \left[\left(\bigcup_{i=1}^{n}\alpha_i\right);\theta\right]\varphi \leftrightarrow \bigwedge_{i=1}^{n}[\alpha_i;\theta]\varphi \leftrightarrow \bigwedge_{i=1}^{n}\left[\bigcap_{j=1}^{n_i}(\alpha_{ij};\theta)\right]\varphi.$$

These equivalences are true by parts (1) and (2) of Lemma 6.1.8. □

We now show that in the formula $[\alpha;\beta]\varphi$ the unions and intersections that occur inside β are both commutative and associative and that the compositions are associative when β is in finite normal form and α is a composition of tests and assignments.

Lemma 6.1.10. *If α is a composition of tests and assignments,*

1. $[\alpha;(\beta\cup\gamma)]\varphi \leftrightarrow [\alpha;(\gamma\cup\beta)]\varphi$,
2. $[\alpha;((\beta\cup\gamma)\cup\delta)]\varphi \leftrightarrow [\alpha;(\beta\cup(\gamma\cup\delta))]\varphi$,
3. $[\alpha;(\beta\cap\gamma)]\varphi \leftrightarrow [\alpha;(\gamma\cap\beta)]\varphi$,
4. $[\alpha;((\beta\cap\gamma)\cap\delta)]\varphi \leftrightarrow [\alpha;(\beta\cap(\gamma\cap\delta))]\varphi$.

Proof.

(1) and (2). Axioms B3.1 and B3.3 imply that both sides of (1) are equivalent to $[(\alpha;\beta)]\varphi\wedge[(\alpha;\gamma)]\varphi$ and that both sides of (2) are equivalent to $[(\alpha;\beta)]\varphi\wedge[\alpha;\gamma]\varphi\wedge[\gamma;\delta]\varphi$.

(3). Axioms B2.4A and B2.2 imply that both sides of (3) are equivalent to $[(\alpha;\beta)\cap(\alpha;\gamma)]\varphi$.

(4). Axioms B2.4A, B2.4B, and B2.2 imply that both sides of (4) are equivalent to $[(\alpha;\beta)\cap(\alpha;\gamma)\cap(\alpha,\delta)]\varphi$. □

Now we show that the conclusions of Lemma 6.1.10 are valid under the more general assumption that α is of the form $\alpha_1 \cap \cdots \cap \alpha_n$ where each α_i is a composition of assignments and tests.

Lemma 6.1.11. *If α is $\alpha_1 \cap \cdots \cap \alpha_n$ where each α_i is a composition of assignments and tests,*

1. $[\alpha; (\beta \cup \gamma)]\varphi \leftrightarrow [\alpha; (\gamma \cup \beta)]\varphi$,
2. $[\alpha; ((\beta \cup \gamma) \cup \delta)]\varphi \leftrightarrow [\alpha; (\beta \cup (\gamma \cup \delta))]\varphi$,
3. $[\alpha; (\beta \cap \gamma)]\varphi \leftrightarrow [\alpha; (\gamma \cap \beta)]\varphi$,
4. $[\alpha; ((\beta \cap \gamma) \cap \delta)]\varphi \leftrightarrow [\alpha; (\beta \cap (\gamma \cap \delta))]\varphi$.

Proof.

1.

$$\left[\left(\bigcap_{i=1}^{n} \alpha_i\right); (\beta \cup \gamma)\right]\varphi \leftrightarrow \left[\bigcap_{i=1}^{n} (\alpha_i; (\beta \cup \gamma))\right]\varphi \tag{5}$$

$$\leftrightarrow \left[(\alpha_1; (\beta \cup \gamma)) \cap \left(\bigcap_{i=2}^{n} (\alpha_i; (\beta \cup \gamma))\right)\right]\varphi \tag{6}$$

$$\leftrightarrow \left[((\alpha_1; \beta) \cup (\alpha_1; \gamma)) \cap \left(\bigcap_{i=2}^{n} (\alpha_i; (\beta \cup \gamma))\right)\right]\varphi \tag{7}$$

$$\leftrightarrow \left[(\alpha_1; \beta) \cap \left(\bigcap_{i=2}^{n} (\alpha_i; (\beta \cup \gamma))\right)\right]\varphi \wedge \left[(\alpha_1; \gamma) \cap \left(\bigcap_{i=2}^{n} (\alpha_i; (\beta \cup \gamma))\right)\right]\varphi. \tag{8}$$

(5) is true by part (2) of Lemma 6.1.8; (6) is the same as (5); (7) is by axiom B3.3, and (8) is by axioms B3.1 and B5.1.

Proceeding in this manner we can reduce the left hand side to

$$\bigwedge_{i=1}^{n} \left[\bigcap_{\lambda_i = \beta, \gamma} (\alpha_i; \lambda_i)\right]\varphi.$$

By using the same reduction sequence we can reduce the right hand side to the last expression.

2. By similar reductions we can show that both sides reduce to

$$\bigwedge_{i=1}^{n} \left[\bigcap_{\lambda_i = \beta, \gamma, \delta} (\alpha_i; \lambda_i)\right]\varphi.$$

3.

$$\left[\left(\bigcap_{i=1}^{n} \alpha_i\right); (\beta \cap \gamma)\right]\varphi \leftrightarrow \left[\bigcap_{i=1}^{n} (\alpha_i; (\beta \cap \gamma))\right]\varphi \tag{9}$$

$$\leftrightarrow \left[\bigcap_{i=1}^{n} ((\alpha_i; \beta) \cap (\alpha_i; \gamma))\right]\varphi. \tag{10}$$

(9) is by part (2) of Lemma 6.1.8 and (10) is by axioms B2.4A, B2.4B, and the commutativity and associativity of $\cap$. By a similar argument we can show that the right hand side reduces to the same value.

4. By using the method of case (3) we can show that both sides reduce to

$$\left[\bigcap_{i=1}^{n}((\alpha_i;\beta)\cap(\alpha_i;\gamma)\cap(\alpha_i;\delta))\right]\varphi. \quad \square$$

Methods of proofs of the previous lemma can be used to prove the following useful identities.

Lemma 6.1.12. *If α is $\bigcap_{i=1}^{n}\alpha_i$ where each α_i is a composition of tests and assignments, then*

$$\left[\left(\bigcap_{i=1}^{n}\alpha_i\right);\left(\bigcup_{j=1}^{n}\beta_j\right)\right]\varphi \leftrightarrow \bigwedge_{i=1}^{n}\left[\bigcap_{j=1}^{n_i}(\alpha_i;\beta_j)\right]\varphi$$

$$\left[\left(\bigcap_{i=1}^{n}\alpha_i\right);\left(\bigcap_{j=1}^{m}\beta_j\right)\right]\varphi \leftrightarrow \left[\bigcap_{i=1}^{n}\bigcap_{j=1}^{m}(\alpha_i;\beta_j)\right]\varphi.$$

Proof. Use reductions similar to case (1) and (3) of Lemma 6.1.11 to reduce the left hand side to the right hand side. $\square$

Lemma 6.1.13. *If α is $\bigcap_{i=1}^{n}\alpha_i$ where each α_i is a composition of tests and assignments and $\beta_j=\bigcap_{k=1}^{l_j}\beta_{j,k}$ where each $\beta_{j,k}$ is a composition of assignments and tests, then*

$$\left[\left(\bigcap_{1=i}^{n}\alpha_i\right);\left(\bigcup_{j=1}^{m}\beta_j\right)\right]\varphi \leftrightarrow \bigwedge_{f\in m^n}\left[\bigcap_{i=1}^{n}\bigcap_{k=1}^{l_{f(i)}}(\alpha_i;\beta_{f(i),k})\right]\varphi.$$

Here the conjunction in the right hand side of the equivalence is over all possible functions f from m to n.

Proof.

$$\left[\left(\bigcap_{i=1}^{n}\alpha_i\right);\left(\bigcup_{j=1}^{m}\right)\beta_j\right]\varphi \leftrightarrow \bigwedge_{f\in m^n}\left[\bigcap_{i=1}^{n}(\alpha_i;\beta_{f(i)})\right]\varphi$$

$$\leftrightarrow \bigwedge_{f\in m^n}\left[\bigcap_{i=1}^{n}\bigcap_{k=1}^{l_{f(i)}}(\alpha_i;\beta_{f(i),k})\right]\varphi.$$

These equivalences are valid by part (1) of Lemma 6.1.12 and repeated application of axiom B2.4B. $\square$

Notice also that $\cap$'s that appear inside each β_j are both associative and commutative. Also the compositions appearing inside each $\beta_{j,k}$ are associative.

Lemma 6.1.14. *If α, β are programs in finite normal form, then the $\cap$'s and $\cup$'s appearing inside α and β are both commutative and associative for $[\alpha;\beta]\varphi$ for any φ. Also compositions appearing inside α and β are associative.*

Proof. Let α be $\cup_{i=1}^{n}\alpha_i$ where each α_i is $\cap_{j=1}^{l_i}\alpha_{ij}$ for some integer l_i, where each α_{ij} is a composition of tests and assignments. The associativity and commutativity of $\cup$ inside α follow from parts (1) and (2) of Lemma 6.1.7. By Lemma 6.1.8 we get

$$[\alpha;\beta]\varphi \leftrightarrow \bigwedge_{i=1}^{n}[\alpha_i;\beta]\varphi.$$

Commutativity and associativity of $\cap$ inside α follow by applying parts (3) and (4) of Lemma 6.1.7 to each $[\alpha_i;\beta]\varphi$. The commutativity and associativity of $\cup$ and $\cap$ inside β follow from Lemma 6.1.11 and part (1) of Lemma 6.1.12.

The associativity of composition follows from applying axioms B1.1B to the right hand side of $[\alpha_i;\beta]\varphi$, after it has been expanded using Lemma 6.1.13. □

The following technical facts will be useful in the derivations of the normal form theorems. They are some auxiliary facts that are used in reductions of programs to their normal forms inside modal operators.

Lemma 6.1.15. *For α free of $\cup$, $\cap$, and $*$,*

1. $[(\alpha;(\beta\cup\gamma))\cap\delta]\varphi \leftrightarrow [(\alpha;(\gamma\cup\beta))\cap\delta]$,
2. $[(\alpha;(\beta\cup(\gamma\cup\rho)))\cap\delta]\varphi \leftrightarrow [(\alpha;((\beta\cup\gamma)\cup\rho))\cap\delta]\varphi$.

Proof. Using axioms B3.4, B5.1, and B3.1 reduces both sides of (1) and (2) to $[(\alpha;\beta\cap\delta]\varphi \wedge [(\alpha;\gamma)\cap\delta]\varphi$ and $[(\alpha;\beta)\cap\delta]\varphi \wedge [(\alpha;\gamma)\cap\delta]\varphi \wedge [(\alpha;\rho)\cap\delta]\varphi$ respectively. □

Lemma 6.1.16. *If α is free of $\cup$, $\cap$, and $*$, then*

$$\left[\alpha;\left(\bigcup_{i=1}^{n}\beta_i\right)\right]\varphi \;\leftrightarrow\; \left[\bigcup_{i=1}^{n}(\alpha;\beta_i)\right]\varphi$$

$$\left[\left(\alpha;\bigcup_{i=1}^{n}\beta_i\right)\cap\delta\right]\varphi \;\leftrightarrow\; \left[\left(\bigcup_{i=1}^{n}(\alpha;\beta_i)\right)\cap\delta\right]\varphi.$$

Proof. The first follows from repeated use of axioms B3.3, B3.1 and the second follows from repeated use of axiom B3.4. □

Lemma 6.1.17.

$$\left[\left(\bigcup_{i=1}^{n}\alpha_i\right)\cap\delta\right]\varphi \leftrightarrow \left[\bigcup_{i=1}^{n}(\alpha_i\cap\delta)\right]\varphi.$$

Proof. By repeated use of axioms B3.1 and B5.1. □

Lemma 6.1.18.

$$\left[\bigcap_{i=1}^{n}\left(\bigcup_{j=1}^{m_i}\beta_{i,j}\right)\right]\varphi \;\leftrightarrow\; \left[\bigcup_{i_1=1,\ldots,i_n=1}^{i_1=m_1,\ldots,i_n=m_n}\left(\bigcap_{j=i}^{n}\beta_{j,i_j}\right)\right]\varphi$$

$$\left[\left(\bigcap_{i=1}^{n}\left(\bigcup_{j=1}^{m_i}\beta_{i,j}\right)\right)\cap\delta\right]\varphi \;\leftrightarrow\; \left[\bigcap_{i_1=1,\ldots,i_n=1}^{i_1=m_1,\ldots,i_n=m_n}\left(\left(\bigcap_{j=i}^{n}\beta_{j,i_j}\right)\cap\delta\right)\right]\varphi.$$

Proof. Follows from repeated applications of Lemma 6.1.17. □

6.2. Finite normal forms for $[\alpha]\varphi$ and $<\alpha>\varphi$ for $*$-free α

In this subsection we show that for each $*$-free programs α we can compute a program α' in finite normal form so that for any proposition φ, $[\alpha]\varphi \leftrightarrow [\alpha']\varphi$ and $<\alpha>\varphi \leftrightarrow <\alpha'>\varphi$ are consequences of our axioms.

Lemma 6.2.1. *For each $*$-free program α, we can compute a $*$-free program α' in finite normal form so that for any proposition φ,*

$$[\alpha]\varphi \leftrightarrow [\alpha']\varphi.$$

Proof. By induction on the structure of α.
Case 0. If α is $x_i \leftarrow t_j$ or $\varphi?$,[8] take α' as α.
Case 1. If α is $\alpha_1 \cup \alpha_2$, then by the inductive hypothesis let α_1', α_2' be the finite normal forms of α_1, α_2 so that

$$[\alpha_1']\varphi \leftrightarrow [\alpha_1]\varphi$$
$$[\alpha_2']\varphi \leftrightarrow [\alpha_2]\varphi.$$

By axiom B3.1 we get

$$[\alpha]\varphi \leftrightarrow [\alpha_1]\varphi \wedge [\alpha_2]\varphi \leftrightarrow [\alpha_1']\varphi \wedge [\alpha_2']\varphi \leftrightarrow [\alpha_1' \cup \alpha_2']\varphi.$$

Now let α' be $\alpha_1' \cup \alpha_2'$.
Case 2. α is $\beta \cap \theta$ where β is $\cup_{i_1}^{n}\beta_i$ and θ is $\cup_{j=1}^{m}\theta_j$ with each β_i being $\cap_{k=1}^{i_n}\beta_{i,k}$ and each θ_i being $\cap_{j=1}^{k}\theta_{i,k}$ where $\beta_{i,k}$ and $\theta_{i,k}$ are compositions of tests and assignments. Then we can derive

$$[\beta \cap \theta]\varphi \leftrightarrow \left[\left(\bigcup_{i=1}^{n}\beta_i\right)\cap\left(\bigcup_{j=1}^{k}\theta_j\right)\right]\varphi \leftrightarrow \left[\bigcup_{i=1,j=1}^{n,m}(\beta_i \cap \theta_j)\right]\varphi$$

by repeated usage of axioms B5.1 and B 2.2. Now let α' be

$$\left(\bigcup_{i=1,j=1}^{n,m}(\beta_i \cap \theta_i)\right).$$

[8] Here φ is a first order formula.

This proof uses associativity and commutativity of $\cup$ and $\cap$ operators occurring inside $\Box$. These were proved in Lemmas 6.1.2 through 6.1.7.

Case 3. α is $\beta;\theta$ where β and θ are as in Case (2). Then

$$[\beta;\theta]\varphi \leftrightarrow \left[\left(\bigcup_{i=1}^{n}\beta_i\right);\theta\right]\varphi \leftrightarrow \bigwedge_{i=1}^{n}[(\beta_i;\theta)]\varphi$$

$$\leftrightarrow \bigwedge_{i=1}^{n}\bigwedge_{f\in m^{n_i}}\left[\bigcap_{k=1,l=1}^{n_i,n_{f(k)}}(\beta_{i,k};\theta_{f(k),l})\right]\varphi \tag{11}$$

$$[\beta;\theta]\varphi \leftrightarrow \left[\bigcup_{i=1}^{n}\bigcup_{f\in m^{n_i}}\left(\bigcap_{k=1,l=1}^{n_i,n_{f(k)}}(\beta_{i,k};\theta_{f(k),l})\right)\right]. \tag{12}$$

(11) is by Lemma 6.1.13, (12) is by repeated applications of axiom B3.1. Take α' as the program on the right hand side of Eq. (12).

Case 4. α is β^*. Then take α' to be β^*. $\Box$

Now we will analyze $<\alpha>\varphi$ for α free of $*$'s. Notice that unlike the axioms for $\Box$, the axioms for $\Diamond$ are compositional; namely,

$$\begin{aligned} <\alpha;\beta>\varphi &\leftrightarrow <\alpha><\beta>\varphi \\ <\alpha\cup\beta>\varphi &\leftrightarrow <\alpha>\varphi\cup<\beta>\varphi \\ <\alpha\cap\beta>\varphi &\leftrightarrow <\alpha>\varphi\wedge<\beta>\varphi \end{aligned}$$

gives us the results analogous to Lemmas 5.2.1 through 5.2.7 and Lemmas 6.1.2 through 6.1.18. The next result is the *normal form* for the diamond operator.

Lemma 6.2.2. *For any $*$-free program α we can compute a program α' in finite normal form so that for any state formula φ,*

$$<\alpha>\varphi \leftrightarrow <\alpha'>\varphi$$

is provable from our axioms.

Proof. We use induction on the inductive definition of programs.

Case 0. α is the assignment $(x_i \leftarrow t)$ or the atomic test $(\theta?)$. Take α' to be α.

Case 1. α is $\alpha_1\cup\alpha_2$. Take α' to be $\alpha_1'\cup\alpha_2'$ where α_1' and α_2' are finite normal forms of α_1 and α_2 respectively.

Case 2. α is $\alpha_1\cap\alpha_2$ where α_1 is $(\bigcup_{i=1}^{l}\beta_i)$ and α_2 is $(\bigcup_{j=1}^{n}\theta_j)$ where each β_i,θ_j are finite intersections of compositions of assignments and tests. Then

$$<\alpha>\varphi \leftrightarrow <\alpha_1\cap\alpha_2>\varphi \leftrightarrow <\left(\bigcup_{i=1}^{l}\beta_i\right)\cap\left(\bigcup_{j=1}^{n}\theta_j\right)>\varphi \tag{13}$$

$$\leftrightarrow <\bigcup_{i=1}^{l}\beta_i>\varphi\wedge<\bigcup_{j=1}^{n}\theta_j>\varphi \tag{14}$$

$$\leftrightarrow < \bigvee_{i=1}^{l} \beta_i > \varphi \wedge \left(\bigvee_{j=1}^{n} < \theta_j > \varphi \right) \tag{15}$$

$$\leftrightarrow \bigvee_{i=1,j=1}^{l,n} (< \beta_i > \varphi \wedge < \theta_j > \varphi) \tag{16}$$

$$\leftrightarrow < \bigcup_{i=1,j=1}^{l,n} (\beta_i \cap \theta_j) > \varphi. \tag{17}$$

(14) follows from axioms D2; (15) follows from axiom D3.1; (16) follows from distributivity; (17) follows from axioms D2.1 and D3.1.

Case 3. α is $(\alpha_1; \alpha_2)$ where α_1 is $\cup_{i=1}^{l} \beta_i$ and α_2 is $\cup_{j=1}^{n} \theta_j$ where β_i is $\cap_{l=1}^{n_i} \beta_{i,l}$ and θ_j is $\cap_{k=1}^{m_j} \theta_{j,k}$ with each $\beta_{i,l}, \theta_{j,k}$ a composition of assignments and tests. Here n_i, m_j depend upon i and j respectively.

$$< \alpha_1; \alpha_2 > \varphi \leftrightarrow < \alpha_1 >< \alpha_2 > \varphi \tag{18}$$

$$\leftrightarrow \bigvee_{i=1}^{l} (< \beta_i >< \alpha_2 > \varphi) \tag{19}$$

$$\leftrightarrow \bigvee_{i=1}^{l} \left(< \bigcap_{l=1}^{n_i} \beta_{i,l} >< \alpha_2 > \varphi \right) \tag{20}$$

$$\leftrightarrow \left(\bigvee_{i=1}^{l} \left(\bigwedge_{l=1}^{n_i} < \beta_{i,l} >< \alpha_2 > \varphi \right) \right) \tag{21}$$

$$\leftrightarrow \bigvee_{i=1}^{l} \left(\bigwedge_{l=1}^{n_i} < \beta_{i,l}; \left(\bigcup_{j=1}^{m} \theta_j \right) > \varphi \right) \tag{22}$$

$$\leftrightarrow \bigvee_{i=1}^{l} \left(\bigwedge_{l=1}^{n_i} \bigwedge_{j=1}^{m} < \beta_{i,l}; \theta_j > \varphi \right) \tag{23}$$

$$\leftrightarrow \bigvee_{i=1}^{l} \left(\bigwedge_{l=1}^{m} < \beta_{i,l}; \bigcap_{k=1}^{m_j} \theta_{j,k} > \varphi \right) \tag{24}$$

$$\leftrightarrow \bigvee_{i=1}^{l} \left(\bigwedge_{l=1}^{n_i} \bigvee_{j=1}^{m} < \bigcap_{k=1}^{m_j} \beta_{i,l}; \theta_{j,k} > \varphi \right) \tag{25}$$

$$\leftrightarrow \bigvee_{i=1}^{l} \left(\bigwedge_{l=1}^{n_j} \bigvee_{j=1}^{m} \bigwedge_{k=1}^{m_j} < \beta_{i,l}; \theta_{j,k} > \varphi \right) \tag{26}$$

$$\leftrightarrow < \bigcup_{i=1}^{n} \bigcup_{f \in m^{n_i}} \left(\bigcap_{k=1,l=1}^{n_i, n_{f(k)}} (\beta_{i,k}; \theta_{f(k),l}) \right) > \varphi. \tag{27}$$

(18) follows from axiom D1.1; (19) follows from repeated applications of axiom D3.1; (20) is (19); (21) follows from repeated applications of axiom D2.1; (22) follows from

axiom D1.1; (23) follows from axioms D3.1 and D3.2; (24) is (23); (25) follows from axioms D2.2; (26) follows from axiom D2.1; (27) follows from axioms D2.1, D3.1, and distributivity. □

6.3. Elimination of $[\alpha]$ *and* $< \alpha >$ *for* $*$*-free* α

In this subsection we show that for α free of $*$'s, $< \alpha > \varphi$ and $[\alpha]\varphi$ are equivalent to a first order formula with no modal operators (provided that φ has none). This implies that CCDL formulas that do not have any $*$'s in them can be replaced by corresponding formulas in pure first order logic. We build up this result in the following series of lemmas. The next lemma shows that $< \alpha > \varphi$ is equivalent to a pure first order formula when α is a composition of assignments and tests.

Lemma 6.3.1. *Let* α *be a composition of tests and assignments. Then we can compute a formula* ψ *with one less occurrence of a modal operator than* $< \alpha > \varphi$ *such that*

$$< \alpha > \varphi \leftrightarrow \psi.$$

Proof. Let α be $\alpha_1; \ldots; \alpha_n$ where each α_i is either a test or an assignment. We use induction on n to prove our claim.
Case 1. α_n is the test $(\theta?)$.

$$< \alpha_1 > \varphi \leftrightarrow < \alpha_1; \ldots; \alpha_{n-1} >< \alpha_n > \varphi \leftrightarrow < \alpha_1; \ldots; \alpha_{n-1} > (\theta \wedge \varphi).$$

by axioms D1.1 and D4. Now the result follows by applying the inductive assumption to $< \alpha_1; \alpha_2; \ldots; \alpha_{n-1} >$.
Case 2. α_n is $(x_i \leftarrow t)$.

$$< \alpha > \varphi \leftrightarrow < \alpha_1; \ldots; \alpha_{n-1} >< \alpha_n > \varphi \leftrightarrow < \alpha_1; \ldots; \alpha_{n-1} > (\exists y(y = t) \wedge \varphi(t/x_i)).$$

By axioms D1.1 and D-trans. Now the result is obtained by applying the inductive assumption to $< \alpha_1; \ldots; \alpha_{n-1} >$. □

Lemmas 6.3.2 and 6.3.3 eliminate $< \alpha >, [\alpha]$ for $*$-free α when these operators are applied to $\Diamond, \Box$-free state formulas.

Lemma 6.3.2. *From any* $*$*-free program* α *and any* $\Diamond, \Box$*-free state formula* φ*, we can compute a* $\Diamond, \Box$*-free state formula* ψ *satisfying*

$$< \alpha > \varphi \leftrightarrow \psi.$$

Proof. We use induction on the inductive definition of program α.
Case 1. α is the assignment $(x_i \leftarrow t)$. Then by axiom D-trans we get

$$< x_i \leftarrow t > \varphi \leftrightarrow \exists y(y = t \wedge \varphi(t/x_i)).$$

Case 2. α is the first order test $(\theta?)$.
Then by axiom D4 we get

$$< \theta? > \varphi \leftrightarrow \theta \wedge \varphi.$$

Case 3. α is $(\alpha_1; \alpha_2)$.
By the inductive assumption there is a modal-free φ_1 satisfying

$$< \alpha_2 > \varphi \leftrightarrow \varphi_1.$$

By axiom D1.1,

$$< \alpha > \varphi \leftrightarrow < \alpha_1 >< \alpha_2 > \varphi \leftrightarrow < \alpha_1 > \varphi_1.$$

Now apply the inductive assumption to $< \alpha_1 > \varphi_1$ and the result follows.
Case 4. α is $\alpha_1 \cup \alpha_2$.
By axiom D3.1,

$$< \alpha > \varphi \leftrightarrow < \alpha_1 > \varphi \vee < \alpha_2 > \varphi.$$

Apply the inductive assumption to $< \alpha_1 > \varphi$ and $< \alpha_2 > \varphi$.
Case 5. α is $\alpha_1 \cap \alpha_2$.
By axiom D2.1,

$$< \alpha > \varphi \leftrightarrow < \alpha_1 > \varphi \wedge < \alpha_2 > \varphi.$$

Apply the inductive assumption to $< \alpha_1 > \varphi$ and $< \alpha_2 > \varphi$. □

Lemma 6.3.3. *From any $*$-free program α and any $\Box, \Diamond$-free state formula φ, we can compute a $\Box, \Diamond$-free state formula θ such that*

$$[\alpha]\varphi \leftrightarrow \theta.$$

Proof. By Lemma 6.2.1 we can compute an $\alpha' = \cup_{i=1}^{n} \alpha_i$ in finite normal form satisfying $[\alpha]\varphi \leftrightarrow [\alpha']\varphi$. Axiom B3.1 implies that

$$[\alpha]\varphi \leftrightarrow \bigwedge_{i=1}^{n} ([\alpha_i]\varphi).$$

For each i there is a $\Box, \Diamond$-free formula ψ_i (by axiom B-Trans) satisfying

$$[\alpha_i]\varphi \leftrightarrow \psi_i$$

and hence

$$[\alpha]\varphi \leftrightarrow \bigwedge_{i=1}^{n} \psi_i. \quad \Box$$

Lemma 6.3.4. *From any $*$-free formula φ we can compute a $\Box, \Diamond$-free formula ψ satisfying*

$$\varphi \leftrightarrow \psi.$$

Proof. Use induction on the definition of formula. The inductive steps corresponding to $\Box, \Diamond$ are given by Lemmas 6.3.2 and 6.3.3. □

6.4. Normal forms of programs

We have shown that any state formula with $*$-free programs can be translated to formulas without modal operators.

Now we look at the general picture. We consider $[\alpha]\varphi$ and $< \alpha > \varphi$ for any program α. Here we define normal forms for arbitrary programs α and show that $[\alpha]\varphi$ and $< \alpha > \varphi$ are equivalent to respectively $[\alpha']\varphi$ and $< \alpha' > \varphi$ where α' is the normal form of α.

Definition 6.4.1 (*Micronormal Forms of Programs*). We say that a program α is in micronormal form if it is of one of the following forms:

1. β^* for some program β.
2. $\beta^*; \gamma$ for some programs β, γ.
3. $\alpha; \beta^*$ for some programs β and α where α is a composition of assignments and tests.
4. $\alpha; \beta^*; \gamma$ for some programs α, β, γ where α is a composition of assignments and tests.

Notice that we usually consider $(\alpha; \beta^*; \gamma)$ to be in micronormal form. (1), (2), and (3) are degenerate cases of (4).

Definition 6.4.2 (*Subnormal and Normal Forms of Programs*). Programs in subnormal and normal forms are defined as follows.

1. We say that program α is in subnormal form if α is of the form $\cap_{i=1}^{n} \alpha_i$ where each α_i is in micronormal form.
2. We say that program α is in normal form if α is of the form $\cup_{i=1}^{n} \alpha_i$ where each α_i is in subnormal form.

Notice that for programs β, θ in normal form and for propositions φ the $\cup$'s and $\cap$'s that appear outside of all $*$ operators in β and θ are both commutative and associative and compositions that appear outside of all $*$ operators are associative in $[\beta \cap \theta]\varphi$ and $[\beta \cup \theta]\varphi$. These are consequences of Lemmas 6.1.2 thorough 6.1.9.

Now we are ready to show that for any program α we can compute a program α' in normal form so that for any proposition φ,

$$\begin{aligned} [\alpha]\varphi &\leftrightarrow [\alpha']\varphi \\ < \alpha > \varphi &\leftrightarrow < \alpha' > \varphi. \end{aligned}$$

Lemma 6.4.3. *From any program α we can compute a program α' in normal form so that for any state formula φ we have $[\alpha]\varphi \leftrightarrow [\alpha']\varphi$.*

Proof. We use induction on the inductive definition of program α.
Case 0. α is the assignment $(x_i \leftarrow t)$ or the test $(\theta?)$.
Then take α' as α.
Case 1. α is $\alpha_1 \cup \alpha_2$.
If α_1' and α_2' are normal forms corresponding to α_1 and α_2, then for any state formula φ we have

$$\begin{aligned} [\alpha_1]\varphi &\leftrightarrow [\alpha_1']\varphi \\ [\alpha_2]\varphi &\leftrightarrow [\alpha_2']\varphi \end{aligned}$$

$$[\alpha_1 \cup \alpha_2]\varphi \leftrightarrow [\alpha_1]\varphi \wedge [\alpha_2]\varphi \leftrightarrow [\alpha_1' \cup \alpha_2']\varphi. \tag{28}$$

Here (28) is by axiom B3.1. Take α' as $\alpha'_1 \cup \alpha'_2$.

Case 2. α is $\alpha_1 \cap \alpha_2$.

Let α_1 be $\cup_{i=1}^{n} \beta_i$ where β_i is $\cap_{k=1}^{n_i} \beta_{i,k}$ and α_2 is $\cup_{i=1}^{m} \theta_i$ where θ_i is $\cap_{k=1}^{m_i} \theta_{i,k}$ with β_i and θ_i programs in subnormal form. Then by methods used in case 2 of the proof of Lemma 6.2.1 we get that

$$[\alpha_1 \cap \alpha_2]\varphi \leftrightarrow \left[\bigcup_{i=1,j=1}^{n,m} (\beta_i \cap \theta_j)\right]\varphi.$$

Now take $\cup_{i=1,j=1}^{n,m}(\beta_i \cap \theta_j)$ as α'.

Case 3. α is $(\alpha_1; \alpha_2)$ where α_1 and α_2 are as in case 2.

Suppose that we assume that (for simplicity) $\beta_{i,k}$ does not have $*$ for $k \leq l_i$ and it has $*$ for $k > l_i$. Then

$$[\alpha]\varphi \leftrightarrow \left[\left(\bigcup_{i=1}^{n} \beta_i\right); \alpha_2\right]\varphi \tag{29}$$

$$\leftrightarrow \bigwedge_{i=1}^{n} [\beta_i; \alpha_2]\varphi \tag{30}$$

$$\leftrightarrow \bigwedge_{i=1}^{n} \left[\bigcap_{j=1}^{l_i} (\beta_{i,j}; \alpha_2)\right]\varphi \tag{31}$$

$$\leftrightarrow \bigwedge_{i=1}^{n} \left[\left(\bigcap_{j=1}^{l_i} (\beta_{i,j}; \alpha_2)\right) \cap \left(\bigcap_{j=l_i+1}^{n_i} \beta_{i,j}; \alpha_2\right)\right]\varphi. \tag{32}$$

Let δ_i denote $\cap_{j=l_i+1}^{n_i}(\beta_{i,j}; \alpha_2)$. Then

$$\leftrightarrow \bigwedge_{i=1}^{n} \left[\left(\bigcap_{j=1}^{l_i} (\beta_{i,j}; \alpha_2)\right) \cap \delta_i\right]\varphi \tag{33}$$

$$\leftrightarrow \bigwedge_{i=1}^{n} \left[\left(\bigcap_{j=1}^{l_i} \left(\beta_{i,j}; \left(\bigcup_{k=1}^{m} \theta_k\right)\right)\right) \cap \delta_i\right]\varphi \tag{34}$$

$$\leftrightarrow \bigwedge_{i=1}^{n} \left[\left(\bigcap_{j=1}^{l_i} \left(\bigcup_{k=1}^{m} (\beta_{i,j}; \theta_k)\right)\right) \cap \delta_i\right]\varphi \tag{35}$$

$$\leftrightarrow \left[\bigcup_{k_i=1,\ldots,k_{l_i}=1}^{k_i=m,\ldots,k_{l_i}=m} \left(\left(\bigcap_{j=1}^{l_i} (\beta_{i,j}; \theta_{k_j})\right) \cap \delta_i\right)\right]\varphi \tag{36}$$

$$\leftrightarrow \left[\bigcup_{i=1}^{n} \bigcup_{k_1=1,\ldots,k_{l_i}=1}^{k_1=m,\ldots,k_{l_i}=m} \left(\bigcap_{j=1}^{l_i} (\beta_{i,j}; \theta_i)\right)\right]\varphi. \tag{37}$$

(30) follows from part (1) of Lemma 6.1.8; (31) follows from part (2) of Lemma 6.1.8; (32) is from commutativity and associativity of $\cap$; (33) is obtained by substituting the value

of δ_i; (34) is obtained from (33) by substituting for α_2; (35) is obtained by Lemma 6.1.16; (36) is obtained by Lemma 6.1.18 and (37) is obtained by repeated applications of axiom B3.1. Take α' as the program appearing in (37).
Case 4. α is α_1^*.
Then α is in the required form. Take α as α'. □

Lemma 6.4.4. *From any program α we can compute a program α' in normal form so that for any state formula φ we have*

$$< \alpha > \varphi \leftrightarrow < \alpha' > \varphi.$$

Proof. By induction on the definition of program α.
Case 0. α is the assignment $(x_i \leftarrow t)$ or the test $(\theta?)$.
Take α' as α.
Case 1. α is $\alpha_1 \cup \alpha_2$.
Take α' as $\alpha_1' \cup \alpha_2'$.
Case 2. α is $\alpha_1 \cap \alpha_2$.
Let α_1 be $\cup_{i=1}^{n}\beta_i$ where β_i is $\cap_{k=1}^{n_i}\beta_{i,k}$ and α_2 is $\cup_{j=1}^{m}\theta_j$ where θ_j is $\cap_{k=1}^{m_j}\theta_{j,k}$ with β_i, θ_j in subnormal form. By the same argument as was used in Case (2) of Lemma 6.2.2 we can show that

$$< \alpha > \varphi \leftrightarrow < \bigcup_{i,j}^{n,m}(\beta_i \cap \theta_j) > \varphi.$$

Take α' as $\cup_{i,j}^{n,m}(\beta_i \cap \theta_j)$.
Case 3. Let α be $\alpha_1; \alpha_2$.
Let α_1, α_2 be as given in case (2). Further assume that $\beta_{i,k}$ does not contain $*$ for $k \le l_i$ and that it contains $*$ for $k > l_i$. Then

$$< \alpha > \varphi \leftrightarrow < \alpha_1; \alpha_2 > \varphi \leftrightarrow < \alpha_1 >< \alpha_2 > \varphi \tag{38}$$

$$\leftrightarrow \bigvee_{i=1}^{n} < \beta_i >< \alpha_2 > \varphi \tag{39}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\left(< \bigcap_{k=1}^{n_i}\beta_{i,k} >< \alpha_2 > \varphi \right) \tag{40}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\bigwedge_{k=1}^{n_i}(< \beta_{i,k} >) < \alpha_2 > \varphi) \tag{41}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\left(\bigwedge_{k=1}^{n_i}(< \beta_{i,k}; \alpha_2 > \varphi) \right) \tag{42}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\left(\bigwedge_{k=1}^{n_i} < \beta_{i,k}; \left(\bigcup_{j=1}^{m}\theta_j\right) > \varphi \right) \tag{43}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\left(\left(\bigwedge_{k=1}^{l_i} < \beta_{i,k}; \left(\bigcup_{j=1}^{m}\theta_j\right) > \varphi\right) \bigwedge < \delta_i{}^{9} > \varphi\right) \tag{44}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\left(\left(\bigwedge_{k=1}^{l_i}\left(\bigvee_{j=1}^{m} < \beta_{i,k}; \left(\bigcap_{t=1}^{m_j}\theta_{j,t}\right) > \varphi\right)\right) \bigwedge < \delta_i > \varphi\right) \tag{45}$$

$$\leftrightarrow \bigvee_{i=1}^{n}\left(\left(\bigwedge_{k=1}^{l_i}\left(\bigvee_{j=1}^{m}\bigwedge_{t=1}^{m_j} < \beta_{i,k}; \theta_{j,t} > \varphi\right)\right) \bigwedge < \delta_i > \varphi\right) \tag{46}$$

$$\leftrightarrow < \bigcup_{i=1}^{n}\bigcup_{k_1=1,\ldots,k_{l_i}=1}^{k_1=m,\ldots,k_{l_i}=m}\left(\left(\bigcap_{j=1}^{l_i}\beta_{i,j}; \theta_{k,j}\right) \cap \delta_i\right) > \varphi. \tag{47}$$

(38) follows from axiom D1.1; (39) follows from repeated applications of axioms D1.1 and D3.1; (40) is a restatement of (39); (41) follows from repeated application of axiom D2.1; (42) follows from axiom D1.1; (43) is expanding α_2 in (42); (44) is a rewriting of (43); (45) follows from axioms D3.2 and D3.1; (46) follows from axioms D2.2 and D2.1; (47) follows from distributivity and axioms D2.1, D3.1.
Now take α' as the program appearing in (47).
Case 4. If α is α_1^*.
Then take α' as α. □

Definition 6.4.5. Suppose that α is any program and φ is any state formula.

- Call $[\alpha']\varphi$ the box normal reduct of $[\alpha]\varphi$ where α' is a normal program obtained by applying the algorithm outlined in Lemma 6.4.3.
- Call $< \alpha' > \varphi$ the diamond normal reduct of $< \alpha > \varphi$ where α' is a normal program obtained by applying the algorithm outlined in Lemma 6.4.4.

Definition 6.4.6 (*Modal Stripped Reducts*).

□ **Reduct** Suppose that α is a $*$-free program in subnormal form. Then we call ψ the *modal stripped reduct* of $[\alpha]\varphi$ if ψ is obtained from $[\alpha]\varphi$ by applying the procedure outlined in axiom B-trans.

◇ **Reduct** Suppose that α is a composition of a finite number of first order tests and assignments. Then we call ψ the modal stripped reduct of $< \alpha > \varphi$ if ψ is obtained from $< \alpha > \varphi$ by applying the algorithm outlined in Lemma 6.2.2.

Notice that in Definition 6.4.6, the nomenclature is mildly misleading. The modal stripped reducts of $[\alpha]\varphi$ or $< \alpha > \varphi$ may not be modal free. They are *modal stripped* in the sense that $[\alpha]$ and $< \alpha >$ have been replaced by a first order construct that does not use modal operators. If φ does not have any other modal operators, then the modal stripped reducts are free of ◇ and □ operators.

[9] δ_i is $\cap_{j=l_i}^{n_i}(\beta_{i,j}; \alpha_2)$.

7. Soundness

Generally, logic papers as well as computer science papers omit proofs of correctness on the grounds that no one would propose incorrect axioms, and their verification is routine anyway. In concurrency, due to the lack of well-developed intuitions in dealing with all the varied possible execution sequences, it is easy to miss cases and produce programs which do not meet their specifications. Indeed, there are many published examples. In CCDL the definitions of the accessibilities, which give rise to possible execution sequences, are delicate and are based on design choices. We want to be sure that the semantics is correctly reflected by the axioms. So we give correctness proofs for the axioms relative to the exact definitions adopted for accessibilities; besides this, for a previous draft, our readers complained that they found it hard to supply these proofs.

7.1. Propositional modal axioms

7.1.1. BOX operator

Associativity of composition

Axioms B1.1A and B1.2A are degenerate cases of B1.1B and B1.2B. The proofs of B1.1B and B1.2B are both similar and due to the following relational identities:

$$\begin{aligned} R((\alpha_0;((\alpha_1;\alpha_2);\alpha_3))\cap\beta) &= R((\alpha_0;((\alpha_1;(\alpha_2;\alpha_3)))\cap\beta) \\ R(((\alpha_1;\alpha_2);\alpha_3)\cap\beta) &= R((\alpha_1;(\alpha_2;\alpha_3))\cap\beta). \end{aligned}$$

These are simple facts of relational algebra. Refer to the next section to see how these imply the axioms.

Associativity and commutativity of $\cap$

This group of axioms (B2.1A through B2.4B) are also valid for the same reason as above; i.e., the corresponding equality holds in relational algebra. As an example we provide a sample proof of the soundness of axiom B2.3A.

Lemma 7.1.1 (*Validity of Axiom B2.3A*).

$$[(\alpha\cap\beta);\gamma]\varphi \leftrightarrow [(\alpha;\gamma)\cap(\beta;\gamma)]\varphi.$$

Proof. Suppose that $s \vdash [(\alpha\cap\beta);\gamma]\varphi$, $s' \geq s$ and $(s', S') \in R((\alpha;\gamma)\cap(\beta,\gamma))$. Then $S' = S'_1 \cup S'_2$, where $(s', S'_1) \in R(\alpha;\gamma)$ and $(s', S'_2) \in R(\beta;\gamma)$ for some $S'_1, S'_2 \subset S(K)$. Hence there exists an $S''_1 \subset S(K)$ with $(s', S''_1) \in R(\alpha)$. For all $s'' \in S''_1$ there is an $S''_{1,s''} \subset S'_1$ satisfying $(s'', S''_{1,s''}) \in R(\gamma)$. There also exists an $S''_2 \subset S(K)$ with $(s', S''_2) \in R(\beta)$ such that for all $s'' \in S''_2$ there is an $S'_{2,s''} \subset S'_1$ satisfying $(s'', S''_{2,s''}) \in R(\gamma)$. Hence $(s', S''_1 \cup S''_2) \in R(\alpha\cap\beta)$ and $(u, S'_1 \cup S'_2) \in R(\gamma)$ for all $u \in S''_1 \cup S''_2$. Hence $(s, S'_1 \cup S'_2) \in R((\alpha\cap\beta);\gamma)$. Hence $u \vdash \varphi$ for all $u \in S'_1 \cup S'_2$.
Conversely, suppose that $s \vdash [(\alpha;\gamma)\cap(\beta;\gamma)]\varphi$. If $s' \geq s$ and $(s', S') \in R((\alpha\cap\beta);\gamma)$,

then there is an $S_1 \cup S_2$ with $(s', S_1) \in R(\alpha)$ and $(s', S_2) \in R(\beta)$ such that for all $s_1 \in S_1$ there is an $S_{s_1} \subset S'$, and for every $s_2 \in S_2$ there is an $S_{s_2} \subset S'$, such that

$$S' = \left(\bigcup_{s_1 \in S_1} S_{s_1} \right) \cup \left(\bigcup_{s_2 \in S_2} S_{s_2} \right).$$

Hence

$$\left(s', \bigcup_{s_1 \in S_1} S_{s_1} \right) \in R(\alpha; \gamma)$$

and

$$\left(s', \bigcup_{s_2 \in S_2} S_{s_2} \right) \in R(\beta; \gamma). \quad \square$$

Associativity and commutativity of $\cup$

We provide sample proofs of B3.1 and B3.2.

Lemma 7.1.2 (*Validity of Axiom B3.1*).

$$[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi.$$

Proof. Suppose that $s \vdash [\alpha \cup \beta]\varphi$, and for some $s' \geq s$ and $S' \subset S(K)$, $(s', S') \in R(\alpha)$. Then $(s', S') \in R(\alpha \cup \beta)$. Hence for all $s'' \in S'$, $s'' \vdash \varphi$.
Conversely, suppose that $s \vdash [\alpha]\varphi \wedge [\beta]\varphi$. If $(s', S') \in R(\alpha \cup \beta)$ for some $s' \geq s$, then $(s', S') \in R(\alpha)$ or $(s', S') \in R(\beta)$. Hence $s'' \vdash \varphi$ for all $s'' \in S'$. $\square$

Lemma 7.1.3 (*Validity of Axiom B3.2*).

$$[(\alpha \cup \beta); \theta]\varphi \leftrightarrow [(\alpha; \theta) \cup (\beta; \theta)]\varphi.$$

Proof. Suppose that $s \vdash [(\alpha \cup \beta); \theta]\varphi$. Let $s' \geq s$ satisfy $(s', S') \in R((\alpha; \theta) \cup (\beta; \theta))$. Then $(s', S') \in R(\alpha; \theta)$ or $(s', S') \in R(\beta; \theta)$. If $(s', S') \in R(\alpha; \theta)$ then there exists an S'' with $(s', S'') \in R(\alpha)$ such that for each $s'' \in S''$ there exists an $S_{s''} \subset S'$ such that $(s'', S_{s''}) \in R(\theta)$ and

$$S' = \bigcup_{s'' \in S''} S_{s''}.$$

Then $(s', S'') \in R(\alpha \cup \beta)$. Hence $(s', S') \in R((\alpha \cup \beta); \theta)$, giving $s \vdash \varphi$ for all $s \in S'$.
Conversely, suppose that $s \vdash [(\alpha; \theta) \cup (\beta; \theta)]\varphi$. If $s' \geq s$ and $(s', S') \in R((\alpha \cup \beta); \theta)$ then there exists an S'' with $(s', S'') \in R(\alpha \cup \beta)$ and

$$S' = \bigcup_{s'' \in S''} S'_{s''}$$

for some $S'_{s''} \subset S'$. Hence $(s', S'') \in R(\alpha)$ or $(s', S'') \in R(\beta)$. Suppose that $(s', S'') \in R(\alpha)$. Then $(s', S') \in R(\alpha; \theta)$. Hence $(s', S') \in R((\alpha; \theta) \cup (\beta; \theta))$, so $s \vdash \varphi$ for all $s \in S'$. $\square$

Distributivity of $\cap$ and $\cup$

We provide a sample proof of axiom B5.1.

Lemma 7.1.4 (*Validity of Axiom B5.1*).

$$[(\alpha \cup \beta) \cap \gamma]\varphi \leftrightarrow [(\alpha \cap \gamma) \cup (\beta \cap \gamma)]\varphi.$$

Proof. We show that $R((\alpha \cup \beta) \cap \gamma) = r((\alpha \cap \gamma) \cup (\beta \cap \gamma))$. Suppose that $(s, S) \in R((\alpha \cup \beta) \cap \gamma)$. Then $S = S_1 \cup S_2$ with $(s, S_1) \in R(\alpha \cup \beta)$ and $(s, S_2) \in R(\gamma)$. Now if $(s, S_1) \in R(\alpha)$ then $(s, S) \in R(\alpha \cap \gamma)$. Hence $(s, S) \in R((\alpha \cap \gamma) \cup (\beta \cap \gamma))$. Conversely, suppose that $(s, S) \in R((\alpha \cap \gamma) \cup (\beta \cap \gamma))$. If $(s, S) \in R(\alpha \cap \gamma)$, then $S = S_1 \cup S_2$ with $(s, S_1) \in R(\alpha)$ and $(s, S_2) \in R(\gamma)$. Hence $(s, S_1) \in R(\alpha \cup \beta)$. Hence $(s, S) \in R((\alpha \cup \beta) \cap \gamma)$, so we have shown that $R((\alpha \cup \beta) \cap \gamma) = R((\alpha \cap \gamma) \cup (\beta \cap \gamma))$. This implies the validity of axiom B5.1. □

The test operator ?

The proof of B6 is very similar to that of D4 and we refer the reader to it.

7.1.2. Diamond operator

Associativity of composition

Lemma 7.1.5 (*Validity of Axiom D1.1*).

$$< \alpha; \beta > \varphi \leftrightarrow < \alpha >< \beta > \varphi.$$

Proof. Suppose that $s \vdash < \alpha; \beta > \varphi$. Then there exists an $s'' \subset S(K)$ with $(s, S'') \in R(\alpha; \beta)$ satisfying $s'' \vdash \varphi$ for all $s'' \in S''$. Hence there exists an $S' \subset S(K)$ with $(s, S') \in R(\alpha)$ satisfying the condition for each $s' \in S'$, there exists an $S_{s'} \subset S''$ with $\cup S_{s'} = S''$ and $(s', S_{s'}) \in R(\beta)$. For all $s'' \in S_s$ we have $s'' \in S''$, and hence $s'' \vdash \varphi$. This implies that $s' \vdash < \beta > \varphi$ for all $s' \in S'$. Hence $s \vdash < \alpha >< \beta > \varphi$.
Conversely if $s \vdash < \alpha >< \beta > \varphi$ then there exists an S' with $(s, S') \in R(\alpha)$ such that $s' \vdash < \beta > \varphi$ for all $s' \in S'$. For each $s' \in S'$ there exists an $S_{s'} \subset S(K)$ such that $(s', S_{s'}) \in R(\beta)$ and $s'' \vdash \varphi$ for all $s'' \in S_{s'}$. Hence

$$\left(s, \bigcup_{s' \in S'} S_{s'}\right) \in R(\alpha; \beta)$$

and $s'' \vdash \varphi$ for all $s'' \in S_{s'}$, and therefore $s \vdash < \alpha; \beta > \varphi$ as required. □

The proof of D1.2 is similar to that of D1.1 and is omitted.

Associativity and commutativity of $\cap$

Lemma 7.1.6 (*Validity of Axiom D2.1*).

$$< \alpha \cap \beta > \varphi \leftrightarrow < \alpha > \varphi \wedge < \beta > \varphi.$$

Proof. $s \vdash < \alpha \cap \beta > \varphi$ if and only if there exists an $S' \subset S(K)$ such that $(s, S') \in R(\alpha \cap \beta)$ satisfying $s' \vdash \varphi$ for all $s' \in S'$. By the definition of $R(\alpha \cap \beta)$ we get $S' = S_1 \cup S_2$, where $(s, S_1) \in R(\alpha)$ and $(s, S_2) \in R(\beta)$ for some $S_1, S_2 \subset S(K)$ such that for all $s_1 \in S_1$, $s_1 \vdash \varphi$ and for all $s_2 \in S_2$, $s_2 \vdash \varphi$. Hence $s \vdash < \alpha > \varphi$ and $s \vdash < \beta > \varphi$.
Conversely, suppose that $s \vdash < \alpha > \varphi \wedge < \beta > \varphi$. Then $s \vdash < \alpha > \varphi$ and $s \vdash < \beta > \varphi$. Hence there exist $S_1, S_2 \subseteq S(K)$ such that $(s, S_1) \in R(\alpha)$ and $(s, S_2) \in R(\beta)$ satisfying $s \vdash \varphi$ for all $s \in S_1 \cup S_2$. Hence $(s, S_1 \cup S_2) \in R(\alpha \cap \beta)$, implying $s \vdash < \alpha \cap \beta > \varphi$ as required. □

The proof of D2.2 is similar to D3.2, and is omitted.

Lemma 7.1.7 (*Validity of Axiom D3.1*).

$$< \alpha \cup \beta > \varphi \leftrightarrow < \alpha > \varphi \vee < \beta > \varphi.$$

Proof. Suppose that $s \vdash < \alpha \cup \beta > \varphi$. Then there exists an $S \subset S(K)$ such that $(s, S) \in R(\alpha \cup \beta)$ and $s' \vdash \varphi$ for all $s' \in S$. Also $R(\alpha \cup \beta) = R(\alpha) \cup R(\beta)$; hence either $(s, S) \in R(\alpha)$ or $(s, S) \in R(\beta)$. Suppose that $(s, S) \in R(\alpha)$. Then $s \vdash < \alpha > \varphi$. If $(s, S) \in R(\beta)$, then $s \vdash < \beta > \varphi$.
Conversely, suppose that $s \vdash < \alpha > \varphi \vee < \beta > \varphi$. Then $s \vdash < \alpha > \varphi$, or $s \vdash < \beta > \varphi$. If $s \vdash < \alpha > \varphi$ then there exists an $S' \subset S(K)$ such that $(s, S) \in R(\alpha)$ and $s' \vdash \varphi$ for all $s' \in S'$. It follows that $(s, S) \in R(\alpha \cup \beta)$. Hence $s \vdash < \alpha \cup \beta > \varphi$, as required. □

Lemma 7.1.8 (*Validity of Axiom D3.2*).

$$< \alpha; (\beta \cup \gamma) > \varphi \leftrightarrow < (\alpha; \beta) \cup (\alpha; \gamma) > \varphi$$

provided that α *is free of* $\cap, \cup$ *and* $*$.

Proof. Suppose that $s \vdash < \alpha; (\beta \cup \gamma) > \varphi$. Then there is an S'' with $(s, S'') \in R(\alpha; (\beta \cup \gamma))$ satisfying $s'' \vdash \varphi$ for all $s'' \in S''$. Hence there is an $S' \subset S(K)$ with $(s, S') \in R(\alpha)$ and for each $s' \in S'$ there is an $S''_{s'}$ satisfying $(s, S''_{s'}) \in R(\beta \cup \gamma)$ and

$$S'' = \bigcup_{s' \in S'} s''_{s'}.$$

If α does not contain $\cap, \cup$, or $*$, then $S' = \{s'\}$ for some single state s'. Hence $(s, \{s'\}) \in R(\alpha)$ and $(\{s'\}, S'') \in R(\beta \cup \gamma)$. Hence $(\{s'\}, S'') \in R(\beta)$ or $(\{s'\}, S'') \in R(\gamma)$. If $(\{s'\}, S'') \in R(\beta)$, then $(s, S'') \in R(\alpha; \beta)$. Hence $(s, S'') \in R((\alpha; \beta) \cup (\alpha; \gamma))$, and $s'' \vdash \varphi$ for all $s'' \in S''$.
Conversely if $s \vdash < (\alpha; \beta) \cup (\alpha; \gamma) > \varphi$, then there is an S' such that $(s, S'') \in R((\alpha; \beta) \cup (\alpha; \gamma))$ satisfying $s'' \vdash \varphi$ for each $s'' \in S''$. Now suppose that $(s, S'') \in (\alpha; \beta)$. Then there is an $S' \subset S(K)$ with $(s, S') \in R(\alpha)$ where for all $s' \in S'$ there exists an $S_{s'} \subset S''$ with $(s', S_{s'}) \in R(\beta)$ and $\bigcup_{s' \in S'} S_{s'} = S''$. Hence $(s', S_{s'}) \in R(\alpha \cup \beta)$ for all $s' \in S'$. Therefore $(s, S'') \in R(\alpha; (\beta \cup \gamma))$. Notice that $s'' \vdash \varphi$ for all $s'' \in S''$, implying $s \vdash < \alpha; (\beta \cup \gamma) > \varphi$. □

Lemma 7.1.9 (*Validity of Axiom D4*).

$$< \theta? > \varphi \leftrightarrow \theta \wedge \varphi.$$

Proof. Suppose that $s \vdash < \theta? > \varphi$. Then there is an $s' \in S(K)$ with $(s, s') \in R(\theta?)$ satisfying $s' \vdash \varphi$ for all $s' \in S'$. Notice that by definition $R(\theta) = \{(s, s) : s \vdash \theta\}$. Hence $S' = \{s'\}$, implying $s \vdash \theta$ and $s \vdash \varphi$. Hence $s \vdash \theta \wedge \varphi$.
Conversely, if $s \vdash \theta \wedge \varphi$, then $s \vdash \theta$ and $s \vdash \varphi$. Now $s \vdash \theta$ implies that $(s, \{s\}) \in R(\theta?)$ and therefore $s \vdash < \theta? > \varphi$. □

7.2. Axioms involving predicate logic

Axioms translating modal operators into first order logic

The following lemma is used later.

Lemma 7.2.1. *If φ does not have any occurrence of $\bar{z}$,*[10] *then $s \vdash \varphi$ if and only if $s(\bar{t}/\bar{z}) \vdash \varphi$.*

Proof. By induction on the structure of φ. □

The proofs of D-trans and B-trans depend upon the following lemmas.

Lemma 7.2.2. *Suppose α is a program that does not contain x_i or any of the variables of t. If $(s, S) \in R(\alpha)$, then $(s(t/x_i),\ S(t/x_i)) \in R(\alpha)$ and conversely, where $S(t/x_i)$ is $\{s'(s'(t)/x_i) : s' \in S\}$. Furthermore, in this case*

1. *$s'(x_j) = s'(s'(t)/x_i)(x_j)$ when x_j is not x_i.*
2. *$S(t'/x_i) = S(t/x_i)(t'/x_i)$ where t' has no x_i in it.*
3. *$s \vdash < \alpha; x_i \leftarrow t' > \psi$ if and only if $s(s(t)/x_i) \vdash < \alpha; x_i \leftarrow t' > \psi$, for any t' having no occurrences of x_i (a consequence of* (2)*).*

Proof. Left as an exercise. □

Lemma 7.2.3. *$s \vdash \varphi(t/x_i)$ if and only if $s(s(t)/x_i) \vdash \varphi(x_i)$ where t is a program term defined in state s.*

Proof. We prove this by induction on the inductive definition of formulas.
Case 0. φ is the atomic formula A.
Then $s \vdash A(t/x_i)$ if and only if $k_s \vdash A(s(t)/x_i)$. This follows from the definition of satisfaction. Then $s(s(t)/x_i) \vdash A(x_i)$ if and only if $k_{s(s(t)/x_i)} \vdash A(s(t)/x_i)$, if and only if $k_s \vdash A(s(t)/x_i)$ (since $k_s = k_{s(s(t)/x_i)}$).
Case 1. φ is one of $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \varphi_1 \rightarrow \varphi_2, \forall y \varphi(y_1), \exists y \varphi_1(y)$.
The proof is simply definition chasing.
Case 2. φ is $< \alpha > \psi$. Suppose that x_i is x_1.
Case 2.1. Suppose that α is not of the special form $(\bar{z} \leftarrow \bar{x}); \beta; (\bar{x} \leftarrow \bar{z})$ where β does not have $\bar{x}$ and ψ does not have $\bar{z}$.
Then by the definition of substitution,

$$s \vdash (< \alpha > \psi)(t/x_i) \quad \text{if and only if} \quad s \vdash (< \bar{z} \leftarrow \bar{x}; \alpha(\bar{z}/\bar{x}); \bar{x} \leftarrow \bar{z} > \psi)(t/x_i).$$

[10] $\bar{z}$ is a vector of program variables and $\bar{t}$ is a vector of program terms.

Suppose that

$$s \vdash < z_1 \leftarrow t; \bar{z}_{n-1} \leftarrow \bar{x}_{n-1}; \alpha(\bar{z}/\bar{x}); \bar{x} \leftarrow \bar{z} > \psi$$

where $\bar{z}$ is a new vector of program variables. Then

$$s \vdash < z_1 \leftarrow t >< \bar{z}_{n-1} \leftarrow \bar{x}_{n-1}; \alpha(\bar{z}/\bar{x}); \bar{x} \leftarrow \bar{z} > \psi$$

if and only if

$$s(s(t)/z_1) \vdash < \bar{z}_{n-1} \leftarrow \bar{x}_{n-1}; \alpha(\bar{z}/\bar{x}); \bar{x} \leftarrow \bar{z} > \psi.$$

(This uses definition of $R(z_1 \leftarrow x_1)$.) By Lemma 7.2.2 the former is equivalent to

$$s(s(t)/z_1)(s(z_1)/x_1) \vdash < (\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z}) > \psi.$$

Notice that $s(s(t)/z_1)(s(z_1)/x_1)$ is $s(s(t)/x_1)(s(x_1)/z_1)$; hence

$$\begin{aligned} s(s(t)/x_1)(x_1/z_1) &\vdash < (\bar{z}_{n-1} \leftarrow \bar{x}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z}) > \psi \\ s(s(t)/x_1) &\vdash < (z_1 \leftarrow x_1); (\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z}) > \psi. \end{aligned}$$

This implies that $s(s(t)/x_1) \vdash < \alpha > \psi$.

Case 2.2. Suppose that α is of the special form $(\bar{z} \leftarrow \bar{x}); \beta; (\bar{x} \leftarrow \bar{z})$ where β does not have x and ψ does not have z. Then apply the same argument as in case 2.1 above to the formula $< (\bar{z} \leftarrow \bar{x}); \beta; (\bar{x} \leftarrow \bar{z}) > \psi$.

Case 3. φ is $[\alpha]\psi$. Suppose that x_i is x_1.

Case 3.1. Suppose that α is not of the special form $(\bar{z} \leftarrow \bar{x}); \beta; (\bar{x} \leftarrow \bar{z})$ where β does not have $\bar{x}$ and ψ does not have $\bar{z}$. We have $[\alpha]\psi \leftrightarrow [(\bar{z} \leftarrow \bar{x}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})]\psi$ by the definition of substitution. We need to show the equivalence of (48) and (49), where

$$s \vdash [(z_1 \leftarrow t); (\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})]\psi \tag{48}$$

$$s(s(t)/x_1) \vdash [(\bar{z}_n \leftarrow \bar{x}_n); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})]\psi. \tag{49}$$

Suppose that (49) is true. Let us have $u \geq s$ with

$$(u, U) \in R((z_1 \leftarrow t); (\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})).$$

Then

$$\begin{aligned} &(u(u(t_1)/z_1), U) \in R((\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x})); (\bar{x} \leftarrow \bar{z})) \\ &(u(u(t_1)/z_1)(u(t_1)/x_1),\ U) \in R((\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})) \qquad (50) \\ &(u(u(t_1)/x_1)(u(t_1)/z_1), U) \in R((\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})) \qquad (51) \\ &(u(u(t_1)/x_1), U) \in R((z_1 \leftarrow x_1); \bar{z}_{n-1} \leftarrow \bar{x}_{n-1}; \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})). \qquad (52) \end{aligned}$$

By (49) for all $v \in U$, $v \vdash \psi$, and so (48) follows.

The reasons for the previous steps are that (50) follows from (49) by Lemma 7.2.2 because $\bar{z}$ has no t in it and (51) follows from (50) because

$$u(u(t_1)/z)(z/x_1) = u((u(t_1)/z_1)(u(t_1)/x_1).$$

Conversely, suppose that (48) is true. Let $u \geq s(s(t)/x_1)$ for u satisfying

$$(u, U) \in R((\bar{z}_n \leftarrow \bar{x}_n); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})).$$

Then u is $u'(u'(t)/x_1)$ for some $u' \geq s$. Hence

$$(u'(u'(t)/x_1), U) \in R((\bar{z}_n \leftarrow \bar{x}_n); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})).$$

By reversing the above argument we get that

$$(u', U) \in R((\bar{z}_1 \leftarrow t); (\bar{z}_{n-1} \leftarrow \bar{x}_{n-1}); \alpha(\bar{z}/\bar{x}); (\bar{x} \leftarrow \bar{z})).$$

Hence (48) implies $v \vdash \psi$ for all $v \in U$, so (49) follows.
Case 3.2. Suppose that α is of the special form $(\bar{z} \leftarrow \bar{x}); \beta; (\bar{x} \leftarrow \bar{z})$, where β does not have $\bar{x}$ and ψ does not have $\bar{z}$. Then apply the same argument as in case 3.1 above to the formula $[(\bar{z} \leftarrow \bar{x}); \beta; (\bar{x} \leftarrow \bar{z})]\psi$. □

Lemma 7.2.4 (*Validity of Axiom D-Trans*).

$$< x_i \leftarrow t > \varphi \leftrightarrow \exists y(y = t) \wedge \varphi(t/x_i).$$

Proof. Suppose that $< x_i \leftarrow t > \varphi$. Then $s \vdash \exists y(y = t)$ and $s(t/x_i) \vdash \varphi$. Hence the previous lemma implies that

$$s \vdash \exists y((y = t) \wedge \varphi(t/x_i)).$$

This argument is reversible. □

The proof of soundness of B-trans requires the following lemma.

Lemma 7.2.5. *Let α be $\alpha_1; \alpha_2; \ldots; \alpha_n$ where each α_i is either a first order test or an assignment. Let $\bar{x}$ be the m-ary vector of all the program variables appearing in α. Define an m-ary program term vector inductively as follows:*

- *$\bar{r}_o$ is $\bar{x}$,*
- *$\bar{r}_{i+1}$ is $\bar{r}_i(t_j(\bar{r}_i/\bar{x})/x_k)$ if α_{i+1} is the assignment $x_k \leftarrow t_j$; else $\bar{r}_i$ if α is a test.*

Finally, let λ be defined as

$$\bigwedge_{i=1}^{n} \{\alpha_i(\bar{r}_i/\bar{x}) : \text{if } \alpha_i \text{ is a test}\}.$$

Then for any state s there is a state s' with $(s, s') \in R(\alpha)$ if and only if

$$s \vdash \lambda \wedge \left(\bigwedge_{i=1}^{n} (\exists y_i = r_i) \right).$$

Furthermore, if either of these are true, then

$$s \vdash \varphi(\bar{r}_n/\bar{x}) \text{ if and only if } s' \vdash \varphi(\bar{x}).$$

Proof. The proof follows from the definition of $R(\alpha)$ and by repeated application of Lemma 7.2.3. □

Lemma 7.2.6 (*Validity of Axiom B-trans*).

$$[(\alpha_{1,1}; \alpha_{1,2}; \ldots; \alpha_{1,n_1}) \cap \cdots \cap (\alpha_{k,1}; \alpha_{k,2}; \ldots; \alpha_{k,n_k})]\varphi \leftrightarrow \psi$$

where each $\alpha_{i,j}$ is either an assignment or a first order test and $\bar{x}$ is the n-ary vector of all program variables appearing in the program and ψ is defined as follows.

1. *Let $\{\bar{r}_{i,j} : i \leq k,\ j \leq n_k\}$ be a double sequence of n-ary program variables defined as follows.*
   - *Let $\bar{r}_{i,0}$ be $\bar{x}$.*
   - *If α_{ij} is $x_m \leftarrow f(\bar{x})$ then let $\bar{r}_{ij}$ be $\bar{r}_{i,j-1}(f(\bar{r}_{i,j-1})/\bar{r}_{i,j-1,m})$ where $\bar{r}_{i,j-1,m}$ is the mth position of $\bar{r}_{i,j-1}$.*
   - *Else let $\bar{r}_{ij}$ be defined as $\bar{r}_{i,j-1}$.*
2. *Define λ_i as $\wedge_{j=1}^{n_i}\{\alpha_{i,j}(\bar{r}_{i,j}) : \alpha_{i,j}$ is a test$\}$.*
3. *Define λ as $\wedge_{i=1}^{n}\lambda_i$.*
4. *Define η as $\bigwedge_{i=1,j=1}^{k,n_k}(\exists\bar{y}_{ij})(\bar{y}_{ij} = \bar{r}_{ij})$.*

Define ψ as $(\eta \wedge \lambda) \rightarrow (\wedge_{j=1}^{k}\varphi(\bar{r}_{j,n_j}/\bar{x}))$.

Proof. We use the notation of the statement of B-trans and the previous lemma. Suppose that

$$s_0 \vdash [(\alpha_{1,1}; \ldots \alpha_{1,n_1}) \cap \cdots \cap (\alpha_{k,1}; \ldots; \alpha_{k,n_k})]\varphi.$$

Take any $s \geq s_0$ and suppose that $s \vdash \eta \wedge \lambda$. Then

$$s \vdash \exists\bar{y}_{i,j}, (\bar{y}_{i,j} = \bar{r}_{i,j}) \wedge (\lambda_1 \wedge \cdots \wedge \lambda_k).$$

By the previous lemma, there are partial states $s_1, \ldots, s_k$ with $(s, s_i) \in R(\alpha_{i,1}; \ldots; \alpha_{i,n_i})$ for each $i \leq k$. Hence

$$(s, \{s_1, \ldots, s_k\}) \in R((\alpha_{i,1}; \ldots; \alpha_{i,n_i}) \cap \cdots \cap (\alpha_{k,1}, \ldots, \alpha_{k,n_k})).$$

Because

$$s_0 \vdash [(\alpha_{1,1}; \ldots; \alpha_{1,n_1}) \cap \cdots \cap (\alpha_{k,1}; \ldots; \alpha_{k,n_k})]\varphi$$

we have that

$$s_i \vdash \varphi(\bar{x})$$

for each $i \leq k$. By the previous lemma,

$$s \vdash \varphi(\bar{r}_{i,n_i}/\bar{x}).$$

Hence

$$s_0 \vdash \psi.$$

Conversely, suppose that $s_0 \vdash \psi$. Take any $s \geq s_0$ and $(s, S) \in R(\alpha)$, where

$$\alpha = (\alpha_{1,1}; \ldots; \alpha_{1,n_1}) \cap \cdots \cap (\alpha_{k,1}; \ldots; \alpha_{k,n_k}).$$

Then let S be $\{s_1, \ldots, s_k\}$ where

$$(s, s_i) \in R(\alpha_i) \text{ and each } \alpha_i \text{ is } (\alpha_{i,1}; \ldots; \alpha_{i,n_i}) \text{ for } i \leq k.$$

Then by the previous lemma,

$$s \vdash \eta \wedge \lambda.$$

Now because $s_0 \vdash \psi$ we have that

$$s \vdash \varphi(\bar{r}_{1,n_1}/\bar{x}) \wedge \cdots \wedge \varphi(r_{k,n_k}/\bar{x}).$$

By the previous lemma,

$$s_i \vdash \varphi(\bar{x}) \text{ for each } i \leq k$$
$$\text{hence } s_0 \vdash [\alpha]\varphi. \quad \square$$

The proofs of the equality axioms are omitted.

8. Tableaux

A tableau proof is always a failed systematic search for a model theoretic counter-example. We suppose a statement false and explore systematically possible constructions of counter-examples. If this systematic search fails in all cases, the resulting tableau, with each branch stopping with a contradiction, is by definition a proof. As described, these tableaux are clearly well-founded trees; the question is: how large? To learn about finite tableau proofs for intuitionistic first order logic, see [5] or [21]. If the semantics of a language has infinitary features, the tableau proofs may be infinite well-founded trees of some transfinite ordinal height. For classical sequential programs of classical dynamic logic, the intended semantics of $[\alpha^*]$, $< \alpha^* >$ already has such infinitary features, even though the language is finitary. The interpretation of $< \alpha^* > \varphi$, $[\alpha^*]\varphi$ already requires branch extension rules giving infinite branching for the signed formulas $T < \alpha^* > \varphi$, $F[\alpha^*]\varphi$. The branch extension rules for $T[\alpha^*]\varphi$, $F < \alpha^* > \varphi$ require only one branch extension, but as we introduce them, require putting an infinite set of formulas on a single branch extension. This also holds for our CCDL tableaux. In summary, we give rules for building tableaux by means of branch extension steps. Our tableaux will be ω-branching trees of height ω with nodes labelled by pairs consisting of a *prefix*, a countable set of signed formulas, and an auxiliary node. This auxiliary node is to represent a partial state over some Kripke model and the set of signed formulas will represent the state formulas that must be forced by this partial state represented by the auxiliary node. All tableau procedures require new constants to be added to the language. So in describing the tableau extension rules we use a countable set of new constants not in CCDL. As far as we know, tableaux have not been used previously in dynamic logic.

Notation. Let σ, ρ be finite sequences of integers. Let S denote a countable set of signed formulas. The node ρ on our tableau with the labeled set S and the auxiliary node σ is denoted by (ρ, σ, S). Then we say that (σ, S) is attached to node ρ. The tableau is a tree of labeled sets. Every node of the tree has no or one or two or ω many immediate extensions, depending on which branch extension rule is used. We write $S \cup \{T(\varphi)\}$, $S \cup \{F(\varphi)\}$ respectively as $S, T(\varphi)$ and $S, F(\varphi)$. We say that μ is a new symbol for an auxiliary node on a branch ρ if no set of signed formulas S with (μ, S) is attached to any node on ρ. We say that a constant d is new on branch ρ if d does not occur in any signed formula in

any set S attached to any node on ρ as a part of its label. We use $<<>>$, a one-to-one function from $\cup_{1=i}^{\infty}\omega^n$ into ω. Then $\rho\hat{}\sigma$ denotes the result of appending the string σ to the right end of string ρ. We denote the tableaux by Ω.

Next we present the relevant branch extension rules and branch modification rules. Any existing node of a tableau may be extended by adding the corresponding extension/modification rule. We also present a list of axioms that may be added to any node of a tableau to extend it.

8.1. Branch extension rules

1. $\boldsymbol{T(\varphi\wedge\psi)}$: If $(\rho,\sigma,(S,T(\varphi\wedge\psi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,T(\varphi),T(\psi)))\in\Omega$.
2. $\boldsymbol{F(\varphi\wedge\psi)}$: If $(\rho,\sigma,(S,F(\varphi\wedge\psi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,F(\varphi)))\in\Omega$ and $(\rho\hat{}<<1>>,\sigma,(S,F(\varphi\wedge\psi)))\in\Omega$.
3. $\boldsymbol{T(\varphi\vee\psi)}$: If $(\rho,\sigma,(S,T(\varphi\vee\psi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,T(\varphi)))\in\Omega$ and $(\rho\hat{}<<1>>,\sigma,(S,T(\varphi)))\in\Omega$.
4. $\boldsymbol{F(\varphi\vee\psi)}$: If $(\rho,\sigma,(S,F(\varphi\vee\psi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,F(\varphi),F(\psi)))\in\Omega$.
5. $\boldsymbol{T(\varphi\rightarrow\psi)}$: If $(\rho,\sigma,(S,T(\varphi\rightarrow\psi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,F(\varphi)))\in\Omega$ and $(\rho\hat{}<<1>>,\sigma,(S,T(\psi)))\in\Omega$.
6. $\boldsymbol{F(\varphi\rightarrow\psi)}$: If $(\rho,\sigma,(S,F(\varphi\rightarrow\psi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma',(\{T(\theta):T(\theta)\epsilon S\},T(\varphi),F(\psi)))\epsilon\Omega$ for a new $\sigma'\geq\sigma$ on the branch.
7. $\boldsymbol{T(\neg\varphi)}$: If $(\rho,\sigma,(S,T(\neg\varphi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,F(\varphi)))\in\Omega$.
8. $\boldsymbol{F(\neg\varphi)}$: If $(\rho,\sigma,(S,F(\neg\varphi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma',(\{T(\theta):T(\theta)\epsilon S\},T(\varphi)))\epsilon\Omega$ for a new $\sigma'\geq\sigma$ on the branch.
9. $\boldsymbol{T([\alpha]\varphi)}$: If $(\rho,\sigma,(S,T([\alpha]\varphi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,T([\alpha']\varphi)))\in\Omega$ where $[\alpha']\varphi$ is the box normal form of $[\alpha]\varphi$.
10. $\boldsymbol{F([\alpha]\varphi)}$: If $(\rho,\sigma,(S,F([\alpha]\varphi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,F([\alpha']\varphi)))\in\Omega$ where $[\alpha']\varphi$ is the box normal form of $[\alpha]\varphi$.
11. $\boldsymbol{T([\alpha\cup\beta]\varphi)}$: If $(\rho,\sigma,(S,T([\alpha\cup\beta])))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,T([\alpha]\varphi\wedge[\beta]\varphi)))\epsilon\Omega$.
12. $\boldsymbol{F([\alpha\cup\beta]\varphi)}$: If $(\rho,\sigma,(S,F([\alpha\cup\beta]\varphi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma(S,F([\alpha]\varphi\wedge[\beta]\varphi)))\in\Omega$.
13. $\boldsymbol{T([\alpha]\varphi)}$ **for subnormal** $\boldsymbol{\alpha}$ **with** $\boldsymbol{*}$**:**
Suppose that $[\alpha]\varphi$ is in box normal form with $*$ and α is $(\alpha_1:\beta_1^*:\gamma_1\cap\ldots\alpha_n;\beta_n^*;\gamma_n)$. If $(\rho,\sigma,(S,T([(\alpha_1;\beta_1^*;\gamma_1)\cap\cdots\cap(\alpha_n;\beta_n^*;\gamma_n)]\varphi)))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma(S,\{T([(\alpha_1;\beta_1^{i_1};\gamma_1)\cap\cdots\cap(\alpha_n;\beta;\gamma_n)]\varphi):i_1,\ldots,i_n\in\omega\}))\in\Omega$.[11]
14. $\boldsymbol{T([\alpha]\varphi)}$ **for subnormal** $\boldsymbol{\alpha}$ **without** $\boldsymbol{*}$**:**
Suppose that α is free of $*$ and $[\alpha]\varphi$ is in subnormal box form. Let ψ be the modal stripped reduct of $[\alpha]\varphi$. If $(\rho,\sigma,(S,T([\alpha]\varphi))\in\Omega$ then $(\rho\hat{}<<0>>,\sigma,(S,T(\psi)))\in\Omega$.
15. $\boldsymbol{F([\alpha]\varphi)}$ **for subnormal** $\boldsymbol{\alpha}$ **with** $\boldsymbol{*}$**:**
Suppose that $[\alpha]\varphi$ is in subnormal box form where α is $(\alpha_1;\beta_1^*;\gamma_1)\cap\cdots\cap(\alpha_n;\beta_n^*;\gamma_n)$ where α_i is a composition of assignments and tests. If $(\rho,\sigma,(S,F([(\alpha_1;\beta_1^*;\gamma_1)\cap\cdots\cap(\alpha_n;\beta_n^*;\gamma_n)]\varphi)))\in\Omega$ then $(\rho\hat{}<<i_1,i_2,\ldots,i_n>>,\sigma',(S',(T[(\alpha_1;\beta_1^i\gamma_1)\cap\cdots\cap$

[11] Notice that one or both of α_i,γ_i may be missing for some i.

$(\alpha_n; \beta_n^{i_n}; \gamma_n)]\varphi))) \in \Omega$, where $S' = \{T(\theta) : T(\theta) \in S\}$ and $\sigma' \geq \sigma$ is new on the branch.

16. $\boldsymbol{F([\alpha]\varphi)}$ **for subnormal** $\boldsymbol{\alpha}$ **without** $*$:
Suppose that $[\alpha]\varphi$ is a subnormal box formula, where α has no $*$. Then let ψ be the modal stripped reduct of $[\alpha]\varphi$. If $(\rho, \sigma, (S, F([\alpha]\varphi)) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, F(\psi))M \in \Omega$.
17. $\boldsymbol{T(< \alpha > \varphi)}$: If $(\rho, \sigma, (S, T(< \alpha > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, T(< \alpha' > \varphi))) \in \Omega$ where $< \alpha' > \varphi$ is the diamond normal form of $< \alpha > \varphi$.
18. $\boldsymbol{F(< \alpha > \varphi)}$: If $(\rho, \sigma, (S, F(< \alpha > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, F(< \alpha' > \varphi))) \in \Omega$ where $< \alpha' > \varphi$ is the diamond normal form of $< \alpha > \varphi$.
19. $\boldsymbol{T(< \alpha \cup \beta > \varphi)}$: If $(\rho, \sigma, (S, T(< \alpha \cup \beta > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, T(< \alpha > \varphi \vee < \beta > \varphi))) \in \Omega$.
20. $\boldsymbol{F(< \alpha \cup \beta > \varphi)}$: If $(\rho, \sigma, (S, F(< \alpha \cup \beta > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, F(< \alpha > \varphi \vee < \beta > \varphi))) \in \Omega$.
21. $\boldsymbol{T(< \alpha \cap \beta > \varphi)}$: If $(\rho, \sigma, (S, T(< \alpha \cap \beta > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, T(< \alpha > \varphi \wedge < \beta > \varphi))) \in \Omega$.
22. $\boldsymbol{F(< \alpha \cap \beta > \varphi)}$: If $(\rho, \sigma, (S, F(< \alpha \cap \beta > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, F(< \alpha > \varphi \wedge < \beta > \varphi)) \in \Omega$.
23. $\boldsymbol{T(< \alpha; \beta^*; \gamma > \varphi)}$ **for micronormal** $\boldsymbol{\alpha; \beta^*; \gamma}$:
Suppose that $< \alpha; \beta^*; \gamma > \varphi$ is a micronormal diamond formula, where α is a composition of assignments and tests. If $(\rho, \sigma, (S, T(< \alpha; \beta^*; \gamma > \varphi))) \in \Omega$ then $(\rho^\frown << i >>, \sigma, (S, T(< \alpha; \beta^i; \gamma > \varphi))) \in \Omega$. Notice that one or both of α, γ may be missing.
24. $\boldsymbol{T(< \alpha > \varphi)}$ **for micronormal** $\boldsymbol{\alpha}$ **without** $*$:
Suppose that $< \alpha > \varphi$ is a micronormal diamond formula, where α does not have any $*$ and ψ is the modal stripped reduct of $< \alpha > \varphi$. If $(\rho, \sigma, (S, T(< \alpha > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, T(\psi))) \in \Omega$.
25. $\boldsymbol{F(< \alpha; \beta^*; \gamma > \varphi)}$**for micronormal** $\boldsymbol{\alpha; \beta^*; \gamma}$:
Suppose that $< \alpha; \beta^*; \gamma > \varphi$ is a micronormal diamond formula, where α is a composition of assignments and tests. If $(\rho, \sigma, (S, F(< \alpha; \beta^*; \gamma > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, \{F(< \alpha; \beta^i; \gamma > \varphi) : i \text{ integer}\})) \in \Omega$.
26. $\boldsymbol{F(< \alpha > \varphi)}$**for subnormal** $\boldsymbol{\alpha}$ **without** $*$:
Suppose that $< \alpha > \varphi$ is a micronormal diamond formula, where α does not have any $*$ and ψ is the modal stripped reduct of $< \alpha > \varphi$. If $(\rho, (S, F(< \alpha > \varphi))) \in \Omega$ then $(\rho^\frown << 0 >>, (S, F(\psi))) \in \Omega$.
27. $\boldsymbol{T(\exists y \varphi(y))}$: If $(\rho, \sigma, (S, T(\exists y \ \varphi(y)))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, T(\varphi(a)), \{F(\psi(a)) : F(\exists y \psi(y)) \in S\})) \in \Omega$ where a is a new constant on the branch.
28. $\boldsymbol{F(\exists y \varphi(y))}$: If $(\rho, \sigma, (S, F(\exists y \varphi(y)))) \in \Omega$ then $(\rho^\frown << 0 >>, (S, \{F(\varphi(e)) :$ where e appears in signed formulas of S$\})) \in \Omega$.
29. $\boldsymbol{T(\forall y \varphi(y))}$: If $(\rho, \sigma, (S, T(\forall y \varphi(y))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma, (S, \{T(\varphi(e)) : e$ appears in signed formulas of S$\})) \in \Omega$.
30. $\boldsymbol{F(\forall y \varphi(y))}$: If $(\rho, \sigma, (S, F(\forall y \ \varphi(y)))) \in \Omega$ then $(\rho^\frown << 0 >>, \sigma', (S, F(\varphi(a)), \{T(\psi) : T(\psi) \in S\}) \in \Omega$, where a is a new constant on the branch and $\sigma' \geq \sigma$ is new on the branch.

8.2. Branch modification and adding axioms

8.2.1. Branch modification rule

If $(\rho, \mu, S), (\rho', \mu', S') \in \Omega$ with $\rho \leq \rho'$ and $\mu \leq \mu'$ then

$$(\rho'^{\frown} << 0 >>, \mu', (S', \{T(\psi) : T(\psi)) \in S\})) \in \Omega.$$

8.2.2. Adding axioms

We may add universal closure equality axioms E1 through E3, the axioms for partial functions fun1 through fun3, and the axiom SV guaranteeing that state variables are defined in each partial state. All instances of other axioms may be added similarly, but our rules have been designed to incorporate them or their consequences into the tableaux.

If $(\rho, \sigma, S) \in \Omega$ then $(\rho^{\frown} << 0 >>, \sigma, (S, T(\varphi))) \in \Omega$ where φ can be any instance of equality axioms.

8.3. Proofs and deductions

Definition 8.3.1 (*Tableau*). A tableau is a tree labelled by sets of signed formulas and an attached auxiliary label and constructed by the branch extension rules, branch modification rules, or the rule on addition of axioms from an initial node with a set of signed formulas with a label $<< 0 >>$ attached to it.

Definition 8.3.2 (*Proof, Deduction*).

Proof: A tableau is a proof if every branch has some node with some label (σ, S) attached to it satisfying $T(\psi), F(\psi) \in S$ for some ψ. A tableau proof is *cut off* below the node that has the occurrence of such a contradiction attached to it.

Proof of φ: A tableau is a proof of φ if the tableau is a proof and the root node has $\{F(\varphi)\}$ attached to it with some label.

Deduction: A tableau deduces φ from $\{\psi_i : i \text{ integer}\}$ if $S = F(\varphi, \{T(\psi_i) : i \in \omega\}$ is attached to its root node as the set of labelled statements with some auxiliary label ρ and the tableau is a proof. □

9. Correctness of tableaux

In this section we establish the correctness of the tableau procedure for the intended semantics. It uses most of the results thus far established. Its length is due to the size of the individual extension steps of the tableau procedure. These steps are much larger than for conventional tableaux, and they reflect natural simplifications in CCDL. To replace them with small steps, one would have to introduce much more elaborate tableaux in which all the accessibility relations are explicitly on the tableaux, which we have avoided. Throughout our proof of correctness of tableaux we assume the semantic equivalence of the formulas of the following kinds.

- $< \alpha > \varphi$ and its diamond normal form.
- $[\alpha]\varphi$ and its box normal form.
- For each α in subnormal form with no $*$'s, $[\alpha]\varphi$ and its modal stripped reduct.
- For each α in micronormal form with no $*$'s, $< \alpha > \varphi$ and its modal stripped reduct.

We notice that these are consequences of our axioms (of which we have demonstrated the semantic validity) using axioms and rules of deductions of first order intuitionistic logic. The soundness of the latter with respect to our semantics of partial states can be verified using standard methods and we omit their repetition here.

In preparation for the verification of the correctness of our tableau procedure, we first provide some preliminary definitions. The first of these provides notation that is necessary for interpreting an ordered set of signed formulas in a partial state of a given Kripke model. The next definition provides notational background necessary to collect sets of signed formulas attached to a path in the tableau for being so interpreted into partial states over a Kripke model.

The method of proof that we use here is an extension of the methods used by Fitting [5] for constant domain predicate modal logics.

For the rest of this section we will be using two different languages. One is our original object language that has constants drawn from a set C. The other is the language with an enlarged set $C \cup P$ of constants and used in the construction of our tableau.

9.1. Interpreting tableau nodes in states

Definition 9.1.1 (*Interpreting a Collection of Sets of CCDL Formulas*). Let Σ be the set of all CCDL formulas over an enlarged set of constants $C \cup P$. Let $U \subset \mathcal{P}(\Sigma)$ be an arbitrary collection of some subsets of Σ. Let $S(K)$ be a set of states over a Kripke model K of the CCDL language that has constants drawn from C. We say that $(\mathcal{F}, \mathcal{C})$ is an interpretation of $(U, \subset)$ in $S(K)$ if

1. $\mathcal{F} : U \to K$ satisfies the monotonicity condition: If $u_1 \subset u_2$ then $\mathcal{F}(u_1) \leq \mathcal{F}(u_2)$ for all $u_1, u_2 \in U$.
2. $\mathcal{C} : U \to \{f \mid f : P \cup X \to D(\mathcal{F}(u))$ is a partial mapping$\}$ satisfying the following properties.
   (a) For each $u \in U, \mathcal{C}(u)$ maps every constant of P that appears in a signed sentence of u to some element of the domain of $\mathcal{F}(u)$.
   (b) $\mathcal{C}(u)(x)$ exists for each $x \in X$.
   (c) If $u_1 \subset u_2$ and $p \in P \cup X$ (i.e. p is a new constant or a program variable) appears in some signed formula of u_1 then $\mathcal{C}(u_1)$ and $\mathcal{C}(u_2)$ assign the same value to p; i.e., $\mathcal{C}(u_1)(p) = \mathcal{C}(u_2)(p)$. Notice that $D(\mathcal{F}(u_1)) \subset D(\mathcal{F}(u_2))$.
   (d) Now we extend $\mathcal{C}(u)$ to all mixed terms mentioned in u (as in Definition 4.2.5) so that all of them are interpreted in $\mathcal{F}(u)$ for each $u \in U$.

Definition 9.1.2 (*Satisfiability of a Collection of Sets of CCDL Formulas*). Let $(U, \subset)$ be a partially ordered set of signed formulas. We say that $(U, \subset)$ is satisfiable if there is a Kripke model K and a set of states $S(K)$ over K and an interpretation $(\mathcal{F}, \mathcal{C})$ of $(U, \subset)$ into $S(K)$ such that:

1. If $T(\varphi) \in u$ then $\mathcal{F}(u) \vdash \varphi$.
2. If $F(\varphi) \in u$ then $\mathcal{F}(u) \nvdash \varphi$.

Definition 9.1.3 (*Interpreting a Branch of a Tableau*). Suppose that μ is a branch on a tableau. For each auxiliary node σ that appears with some set of signed formulas on

this branch let $U_{\sigma,\mu} = \cup\{S : (\sigma, S)$ appear on some node of $\mu\}$. Now define $W_{\sigma,\mu}$ as $\cup_{\sigma' \leq \sigma} U_{\sigma',\mu}$. Let $V(\mu)$ be the set of all such $W_{\sigma,\mu}$ and let $\Gamma(\mu)$ be the ordered structure $(V_\mu, \subset)$. We say that the branch μ is satisfiable if $\Gamma(\mu)$ is satisfiable in the sense of Definition 9.1.2.

9.2. Validity of tableaux

Lemma 9.2.1 (*Correctness of Tableau Extension Rules*). *Suppose that μ is a finite branch of a tableau where $\Gamma(\mu)$ is satisfiable. Suppose that μ is extended by applying branch extension, branch modification, or the addition of axiom rules. Then there is an extension μ' of μ on the tableau where $\Gamma(\mu')$ is satisfiable.*

Proof. Since the tableau extensions of μ depend upon the choice of the extension rule used to extend μ, the choice of μ' also depends on this choice too. The branch extension rules for the usual intuitionistic connectives are those given in [21]. We omit those cases which are analogous to standard proofs of tableau correctness for pure first order intuitionistic logic with equality (see [21,5] for these). We present only those cases that involve the modal connectives.

1. $\boldsymbol{T([\alpha]\varphi)}$ **and** $\boldsymbol{F([\alpha]\varphi)}$: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T([\alpha]\varphi, S)) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma(T([\alpha']\varphi), S)) \in \Omega$ where $[\alpha']\varphi$ is the box normal reduct of $[\alpha]\varphi$. By satisfiability of $\Gamma(\mu)$ in $S(K)$ by $(\mathcal{F}, \mathcal{C})$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha]\varphi$. By semantic equisatisfiability of $[\alpha]\varphi$ and $[\alpha']\varphi$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha']\varphi$. Take μ' as $\mu^\frown << 0 >>$. Now $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ appearing on μ'. A similar argument applies if $F([\alpha]\varphi) \in S_\sigma$ and we add $F([\alpha']\varphi)$ to it to get the new tableau node.
2. $\boldsymbol{T([\alpha \cup \beta]\varphi)}$ **and** $\boldsymbol{F([\alpha \cup \beta]\varphi)}$: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T([\alpha \cup \beta]\varphi), S)) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, (T([\alpha]\varphi \wedge [\beta]\varphi), S)) \in \Omega$. By satisfiability of $\Gamma(\mu)$ in $S(K)$ by $(\mathcal{F}, \mathcal{C})$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha \cup \beta]\varphi$. By the equivalence of $[\alpha \cup \beta]\varphi$ and $[\alpha]\varphi \wedge [\beta]\varphi$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha]\varphi \wedge [\beta]\varphi$. Take μ' as $\mu^\frown << 0 >>$. Now $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ. A similar argument applies if $F([\alpha \cup \beta]\varphi) \in S_\sigma$ and if we add $F([\alpha]\varphi \wedge [\beta]\varphi)$ to S_σ to get the new tableau node.
3. $\boldsymbol{T([\alpha]\varphi)}$ **for subnormal** $\boldsymbol{\alpha}$:
   (a) Suppose that $[\alpha]\varphi$ is a subnormal box formula α has no $*$ in it and ψ is the modal stripped reduct of $[\alpha]\varphi$. Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T([\alpha]\varphi), S)) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, (T(\psi), S)) \in \Omega$. By satisfiability of $\Gamma(\mu)$ we have that $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha]\varphi$ and by semantic equisatisfiability of $[\alpha]\varphi$ and ψ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash \psi$.
   (b) Suppose that $\alpha = (\alpha_1; \beta_1^*; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^*; \gamma_n)$ is a subnormal box formula and each α_i is a composition of assignments and tests. Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T([\alpha]\varphi), S)) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, S') \in \Omega$ where S' is $S \cup \{T([(\alpha_1; \beta_1^{i_1}; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^{i_n}; \gamma_n)]\varphi) : i_1 \cdots i_n$ integer$\}$. Then by satisfiability of $\Gamma(\mu)$ we get $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha]\varphi$. Hence $\mathcal{F}(U_{\sigma,\mu}) \vdash [(\alpha_1; \beta_1^{i_1}; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^{i_n}; \gamma_n)]\varphi$ for any integers $i_i, \ldots, i_n$. Take μ' as $\mu^\frown << 0 >>$. Hence

$\Gamma(\mu')$ is satisfiable, by $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ mentioned on the branch μ'. This argument applies in the degenerate cases where one or both of α_i, γ_i may be missing.

4. ***F*([α]φ) for subnormal α**:
   (a) Suppose that $[\alpha]\varphi$ is a subnormal box formula, α has no $*$'s in it and ψ is the modal stripped reduct of $[\alpha]\varphi$. Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T([\alpha]\varphi), S)) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, (T(\psi), S)) \in \Omega$. Then by satisfiability of $\Gamma(\mu)$ we get $\mathcal{F}(U_{\sigma,\mu}) \vdash [\alpha]\varphi$ and by equisatisfiability of $[\alpha]\varphi$ and ψ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash \psi$. Take μ' as $\mu^\frown << 0 >>$. Hence $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ that are mentioned on branch μ'.
   (b) Suppose that α is $(\alpha_1; \beta_1^*; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^*; \gamma_n)$ and $[\alpha]\varphi$ is a normal box formula. Suppose also that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (F([\alpha]\varphi), S)) \in \Omega$, and for all integers $j_1, \ldots, j_n$, $(\mu^\frown << j_1, \ldots, j_n >>, \rho, S') \in \Omega$ where $S' = (F([(\alpha_1; \beta_1^{j_1}; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^{j_n}; \gamma_n)]\varphi) \cup \{T(\psi) : T(\psi) \in S_\sigma\}$ and $\rho \geq \sigma$ is new on μ. Then by satisfiability of $\Gamma(\mu)$ we get $\mathcal{F}(U_{\sigma,\mu}) \vdash [(\alpha_1; \beta_1^*; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^*; \gamma_n)]\varphi$. There is a partial state of $S(K)$, say $s \geq \mathcal{F}(U_{\sigma,\mu})$, and some integers $i_1, \ldots, i_n$ satisfying $s \vdash [(\alpha_1; \beta_1^{i_1}; \gamma_1) \cap \cdots \cap (\alpha_n; \beta_n^{i_n}; \gamma_n)]\varphi$. Let μ' be $\mu^\frown << i_1, \ldots, i_n >>$, $\mathcal{F}(U_{\rho,\mu'}) = (k, s)$ and let $\mathcal{C}(U_{\rho,\mu'})$ map the same values to the same individuals that $\mathcal{C}(U_{\sigma,\mu})$ mapped them into. Now $\Gamma(\mu')$ becomes satisfiable, by the mapping $\mathcal{F}(U_{\lambda\mu'}) = \mathcal{F}(U_{\lambda,\mu})$ for all λ mentioned on the branch μ'. Notice that one or both of α_i, γ_i may be missing for some i.
5. ***T*(< α > φ) and *F*(< α > φ)**: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T([\alpha]\varphi, S))) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, (T(< \alpha' > \varphi), S)) \in \Omega$. Here $< \alpha' > \varphi$ is the diamond normal reduct of $< \alpha > \varphi$. By the satisfiability of $\Gamma(\mu)$ we have that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha > \varphi$. Then by equisatisfiability of $< \alpha > \varphi$ and $< \alpha' > \varphi$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha' > \varphi$. Take μ' as $\mu^\frown << 0 >>$ and $\mathcal{F}(U_{\lambda,\mu'}) = \mathcal{F}(U_{\lambda,\mu})$ for all λ mentioned on branch μ'. Then this mapping gives us the satisfiability of $\Gamma(\mu')$. A similar argument applies if $F(< \alpha > \varphi) \in S$ and we added $F(< \alpha' > \varphi)$ to S to make the new tableau rule.
6. ***T*(< α ∩ β > φ) and *F*(< α ∩ β > φ)**: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T(< \alpha \cap \beta > \varphi), S))) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, (T(< \alpha > \varphi \wedge < \beta > \varphi, S)) \in \Omega$. By satisfiability of $\Gamma(\mu)$ in some $S(K)$ by $(\mathcal{F}, \mathcal{C})$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha \cap \beta > \varphi$. By equisatisfiability of $< \alpha \cap \beta > \varphi$ and $< \alpha > \varphi \wedge < \beta > \varphi$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha > \varphi \wedge < \beta > \varphi$. Take μ' as $\mu^\frown << 0 >>$. Now $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\sigma,\mu'}) = \mathcal{F}(U_{\sigma,\mu})$ for all ρ mentioned on the branch μ'. A similar argument applies if $F(< \alpha \cap \beta > \varphi) \in S$ and if we added $F(< \alpha > \varphi \Lambda < \beta > \varphi)$ to obtain a new node of the tableau.
7. ***T*(< α ∪ β > φ) and *F*(< α ∪ β > φ)**: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T(< \alpha \cup \beta > \varphi), S))) \in \Omega$, and $(\mu^\frown << 0 >>, \sigma, (T(< \alpha > \varphi \vee < \beta > \varphi), S_\sigma)) \in \Omega$. By satisfiability of $\Gamma(\mu)$ in some $S(K)$ by some $(\mathcal{F}, \mathcal{C})$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha \cup \beta > \varphi$. By equisatisfiability of $< \alpha \cup \beta > \varphi$ and $< \alpha > \varphi \vee < \beta > \varphi$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha > \varphi \vee < \beta > \varphi$. Take μ' as $\mu^\frown << 0 >>$. Then $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ mentioned on the branch μ'. A similar argument applies if $F(< \alpha \cup \beta > \varphi) \in S$ and we add $F(< \alpha > \varphi \vee < \beta > \varphi)$ to get a new node of the tableau.

8. ***T*(< α > φ) for α**: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T(< \alpha > \varphi), S))) \in \Omega$, α is a micronormal program and $< \alpha > \varphi$ is a micronormal diamond formula.

   (a) Suppose that α is free of $*$'s and ψ is the modal stripped diamond reduct of $< \alpha > \varphi$. Also assume that $(\mu^\frown << 0 >>, \sigma, (T(\psi), S))) \in \Omega$. By satisfiability of $\Gamma(\mu)$ we have that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha > \varphi$. Hence by equisatisfiability of $< \alpha > \varphi$ and ψ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash \psi$. Take μ' as $\mu^\frown << 0 >>$. Then $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for each ρ mentioned on branch μ'.
   (b) Suppose that α is $\alpha_1; \beta_1^*; \gamma_1$ and $T(< \alpha_1; \beta_1^*; \gamma_1 > \varphi) \in S$. Then $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha_1; \beta_1^*; \gamma_1 > \varphi$. By semantic interpretation of β_1^*, we have that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha_1; \beta_1^i; \gamma_1 > \varphi$ for some i. Now by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ we get satisfiability of $\Gamma(\mu')$ for $\mu' = \mu^\frown << i >>$. Notice that one or both of α_1, γ_1 may be missing.

9. ***F*(< α > φ for micronormal α**: Suppose that $\Gamma(\mu)$ is satisfiable, $(\mu, \sigma, (T(< \alpha \cup \beta > \varphi), S)) \in \Omega$, and α is a micronormal program.

   (a) α has no $*$. This is similar to the verification of $T(< \alpha > \varphi)$ for micronormal α.
   (b) Suppose that α is $\alpha_1; \beta_1^*; \gamma_1$. The degenerate cases include the possibility of one or both of α_1, γ_1 being missing. By satisfiability of $\Gamma(\mu)$ we get that $\mathcal{F}(U_{\sigma,\mu}) \vdash < \alpha_1, \beta_1^*; \gamma_1 > \varphi$. Hence $\mathcal{F}(U_{\sigma,\mu'}) \vdash < \alpha_1; \beta_1^i; \gamma_1 > \varphi$ for each i. Take μ' as $\mu^\frown << 0 >>$. Then $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ mentioned on the branch μ'.

10. **Addition of axioms:** If μ is finite branch of the tableau with $\Gamma(\mu')$ satisfiable, suppose for the stages of addition of axioms that we can take μ' to be $\mu^\frown << 0 >>$. In all these cases $\Gamma(\mu')$ is satisfiable, by taking $\mathcal{F}(U_{\rho,\mu'}) = \mathcal{F}(U_{\rho,\mu})$ for all ρ. This is because of the validity of the equality axioms, uniqueness of a function value (when it exists), and the requirement for an argument of a function to be defined when a value is defined. The addition of $\exists y(y = x_i)$ is true because of the definition of a state; namely, in every state every program variable is assigned a value.

11. **Branch modification rule:** Application of the branch modification rule is justifiable by the monotonicity of forcing. □

Lemma 9.2.2 (*Correctness of Tableau Deductions*). *If φ is tableau deducible from* $\{\psi_i : i$ *integer*$\}$ *then* $s \vdash \varphi$ *when* $s \vdash \psi_i$ *for all* i, *for any state* $s \in S(K)$ *for any set of states S(K) over any Kripke model K*.

Proof. If not, then there is a state $s \in S(K)$ in a Kripke model K in which $s \vdash \psi_i$ for all i, but $s \nvdash \varphi$. This implies that the tableau with a single node $(<< 0 >>, 0, \{F\varphi\} \cup \{T(\psi_i) : i$ integer $\})$ is satisfiable. The correctness theorem for the tableau extension rule implies that this single node tableau is extendable to a tableau (say) Ω having a branch (say μ) such that $\Gamma(\mu)$ is satisfiable.

Since φ is deducible from $\{\psi_i : i$ integer $\}$, there is some auxiliary node σ and a set S of signed formulas (μ, σ, S) on Ω with some formula θ satisfying $T(\theta), F(\theta) \in S$. This contradicts the satisfaction of $U_{\sigma,\mu}$ by any state in $S(K)$. □

10. Consistency properties and models

We define the notion of consistency property and using a partial ordering on formulas show that a consistency property can be used to produce a model.

10.1. Consistency property

Definition 10.1.1 (*Consistency Property*). Let Σ be the set of signed formulas of CCDL over an enlarged set of constants $C \cup P$. Call $(U, \leq)$ a consistency property if $U \subseteq \mathcal{P}(\Sigma)$ and $\leq$ is a partial ordering on U satisfying the following properties.

1. (a) If $S \leq S'$ and $T\varphi \in S$ then $T\varphi \in S'$.
   (b) For no φ do we have $T\varphi \in S$ and $F\varphi \in S$ for any $S \in U$.
2. (a) If $T(\varphi \wedge \theta) \in S$ then $T(\varphi) \in S$ and $T(\theta) \in S$. If $F(\varphi \wedge \theta) \in S$ then either $F(\varphi) \in S$ or $F(\theta) \in S$.
   (b) If $T(\varphi \vee \theta) \in S$ then either $T(\varphi) \in S$ or $T(\theta) \in S$. If $F(\varphi \vee \theta) \in S$ then $F(\varphi) \in S$ and $F(\theta) \in S$.
   (c) If $T(\varphi \to \theta) \in S$ then either $F(\varphi) \in S$ or $T(\theta) \in S$. If $F(\varphi \to \theta) \in S$ then there is some $S' \in U$ with $S' \geq S$ and $T(\varphi), F(\theta) \in S'$.
   (d) If $T(\neg\varphi) \in S$ then $F(\varphi) \in S$. If $F(\neg\varphi) \in S$ then there is some $S' \in U$ with $S' \geq S$ and $T(\varphi) \in S'$.
   (e) If $T(\forall y \varphi(y)) \in S$ then for all constants e of the extended language that appear in some formula of S, $T(\varphi(e)) \in S$. If $F(\forall y \varphi(y)) \in S$ then there is some $S' \in U$ with $S' \geq S$ and $F(\varphi(e)) \in S'$ for some constant e of the extended language.
   (f) If $T(\exists y \varphi(y)) \in S$ then $T(\varphi(e)) \in S$ for some constant e of the extended language. If $F(\exists y \varphi(y)) \in S$ then $F(\varphi(e))$ for constants e that appear in any formula of S.
   (g) If $T(< \alpha > \varphi) \in S$ then $T(< \alpha_1 > \varphi) \in S$. If $F(< \alpha > \varphi) \in S$ then $F(< \alpha_1 > \varphi) \in S$ where $< \alpha_1 > \varphi$ is the diamond normal reduct of $< \alpha > \varphi$.
   (h) If $T(< \alpha \cup \beta > \varphi) \in S$ then $T(< \alpha > \varphi \vee < \beta > \varphi) \in S$. If $F(< \alpha \cup \beta > \varphi) \in S$ then $F(< \alpha > \varphi \vee < \beta > \varphi) \in S$.
   (i) If $T(< \alpha \cap \beta > \varphi) \in S$ then $T(< \alpha > \varphi \wedge < \beta > \varphi) \in S$. If $F(< \alpha \cap \beta > \varphi) \in S$ then $F(< \alpha > \varphi \wedge < \beta > \varphi) \in S$.
   (j) If α is a composition of assignments and first order tests and $T(< \alpha; \beta^*; \gamma > \varphi) \in S$ then $T(< \alpha; \beta^i; \gamma > \varphi) \in S$ for some i. Also if $F(< \alpha; \beta^*; \gamma > \varphi) \in S$ then $F(< \alpha; \beta^i; \gamma > \varphi) \in S$ for all i. These conditions must hold for the degenerate cases of micronormal programs where one or both of α, γ may be missing.
   (k) For $< \alpha > \varphi$ a micronormal diamond formula where α is free of $*$, if $T(< \alpha > \varphi) \in S$ then $T(\psi) \in S$. If $F(< \alpha > \varphi) \in S$ then $F(\psi) \in S$ where ψ is the modal stripped reduct of $< \alpha > \varphi$.
   (l) If $T([\beta]\varphi) \in S$ then $T([\beta']\varphi) \in S$. If $F([\beta]\varphi) \in S$ then $F([\beta']\varphi) \in S$ where $[\beta']\varphi$ is the normal reduct of $[\beta]\varphi$.
   (m) If $T([\alpha \cup \beta]\varphi) \in S$ then $T([\alpha]\varphi \wedge [\beta]\varphi) \in S$. If $F([\alpha \cup \beta]\varphi) \in S$ then $F([\alpha]\varphi \wedge [\beta]\varphi) \in S$.
   (n) Let $\alpha = (\beta_1; \gamma_1^*; \theta_1) \cap \cdots \cap (\beta_n; \gamma_n^*; \theta_n)$ where α is a normal program. If $T([\alpha]\varphi) \in S$ then

$$[(\beta_1; \gamma_1^{i_1}; \theta_1) \cap (\beta_2; \gamma_2^{i_2}; \theta_2) \cap \cdots \cap (\alpha_n; \beta_n; \gamma_n^{i_n}); \theta_n]\varphi \in S$$

for all integers $i_1, \ldots, i_n$. If $F([\alpha]\varphi) \in S$ then for some $S' \geq S$ and for some integers $i_1, \ldots, i_n$,

$$F([(\beta_1; \gamma_1^{i_1}; \theta_1) \cap \cdots \cap (\beta_n; \gamma_n^{i_n}; \theta_n)]\varphi) \in S'.$$

Note that one or both of β_i, θ_i may be missing for some i.

(o) For α a subnormal program that is free of $*$'s, If $T([\alpha]\varphi) \in S$ then $T(\psi) \in S$. If $F([\alpha]\varphi) \in S$ then $F(\psi) \in S$ where ψ is the modal stripped reduct of $[\alpha]\varphi$.

(p) The universal closures of all instances of equality axioms are in each $S \in U$.

10.2. Towards the model existence theorem

This theorem states that one can construct a model from a consistency property. Once this construction is done we need to verify its correctness by using a well ordering on the formulas. The current subsection develops this well ordering.

Definition 10.2.1 (*The $*$-level of a Program*). (Defined by induction on the structure of programs.)

- If γ is $x \leftarrow t$ or $\varphi?$ then $L(\gamma) = 0$.
- If γ is $\alpha \cap \beta$, $\alpha \cup \beta$ or $\alpha; \beta$ then

$$L(\gamma) = \max\{L(\alpha), L(\beta)\}.$$

- If γ is α^* then $L(\gamma) = L(\alpha) + 1$.

The next definition presents a measure that keeps track of the structure of the embedding of the $*$'s in a program. To this end we define the $*$-depth of a program as a finite sequence of integers. We review a few basic notations. Let ω^ω be the set of finite sequences of integers. If $f, g \in \omega^\omega$ then we denote their pointwise sum by $f + g$. We take $<$ to be the dictionary order on ω^ω. Let i_n (for $n \in \omega$) denote the characteristic function of n in ω^ω. Let 0_0 denote the element of ω^ω that is 0 for each $n \in \omega$.

Definition 10.2.2 (*Depth of a Program*). (Defined by induction on the inductive definition of programs and takes a value in ω^ω.)

- $d(x \leftarrow t), d(\varphi?) = 0_0$.
- $d(\alpha; \beta) = d(\alpha) + d(\beta)$.
- $d(\alpha \cup \beta) = d(\alpha \cap \beta) = \max\{d(\alpha), d(\beta)\}$.
- $d(\alpha^*) = d(\alpha) + 1_{L(\alpha^*)}$.

The following results are immediate about the depths of programs.

Lemma 10.2.3 (*Some Basic Facts About Depths*).

1. $d((\alpha \cup \beta); \gamma) = d((\alpha; \gamma) \cup (\beta; \gamma))$.
2. $d((\alpha \cap \beta); \gamma) = d((\alpha; \gamma) \cap (\beta; \gamma))$.
3. $d(\gamma; (\alpha \cup \beta)) = d((\gamma; \alpha) \cup (\gamma; \beta))$.
4. $d(\gamma; (\alpha \cap \beta)) = d((\gamma; \alpha) \cap (\gamma; \beta))$.
5. $d((\cup_{i=1}^n \alpha_i) \cap (\cup_{j=1}^m \beta_j)) = d(\cup_{i=1, j=1}^{n,m} (\alpha_i \cap \beta_j))$.

6. *If* $< \alpha > \varphi$ *is the diamond normal form of* $< \alpha' > \varphi$ *and* $[\beta]\varphi$ *is the box normal form of* $[\beta']\varphi$ *then*
   - $d(\alpha) = d(\alpha')$,
   - $d(\beta) = d(\beta')$.
7. $d(\cap_{i=1}^{n}(\alpha_i; \beta_i^*; \gamma_i)) > d(\cap_{i=1}^{n}(\alpha_i; \beta_i^{k_i}; \gamma_i))$ *for any integers* $k_1, \ldots, k_n$. *This is true even in the degenerate cases where one or both of* α_i, γ_i *may be missing.*

Proof. The proofs of (1) through (5) are definition chasing; (6) is proved following the reduction sequence outlined in the proofs of normal form theorems. The proof of (7) is again definition chasing. □

Definition 10.2.4 (*$*$-depth of a Formula*). (Defined by induction on the inductive definition of formulas and takes a value in ω^ω.)

1. If φ is an atomic formula then $SD(\varphi) = 1_o$.
2. If φ is $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \varphi_1 \rightarrow \varphi_2$ then
$$SD(\varphi) = \max\{SD(\varphi_1), SD(\varphi_2)\}.$$
3. If φ is $\neg\varphi_1, \forall y\varphi_1(y), \exists y\varphi_1(y)$ then
$$SD(\varphi) = SD(\varphi_1).$$
4. If φ is $< \alpha > \varphi$ or $[\alpha]\varphi$ then
$$SD(\varphi) = d(\alpha)$$
where $d(\alpha)$ is the $*$-depth of α.

The concept of *modal depth* of a formula, as given in the next definition, provides a measure of the number of modal operators embedded in the formula.

Definition 10.2.5 (*Modal Depth*). (Defined by induction and takes an integer value.)

1. If φ is an atomic formula then $MD(\varphi) = 0$.
2. If φ is $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \varphi_1 \rightarrow \varphi_2$ then $MD(\varphi) = \max\{MD(\varphi_1), MD(\varphi_2)\}$.
3. If φ is $\neg\varphi_1, \forall y\varphi_1(y), \exists y\varphi_1(y)$ then $MD(\varphi) = MD(\varphi_1)$.
4. If φ is $< \alpha > \varphi$ or $[\alpha]\varphi$ then $MD(\varphi) = 1 + MD(\varphi)$.

The following facts are needed in the subsequent development.

Lemma 10.2.6 (*Some Basic Facts About SD and MD*).

- *For any formula* φ, $MD(\varphi) \in \omega$ *and* $SD(\varphi) \in \omega^\omega$.
- *If* y *is a free logic variable and* c *is any constant then*
$$MD(\varphi) = MD(\varphi(c/y))$$
$$SD(\varphi) = SD(\varphi(c/y))$$
- $SD(< \alpha \cup \beta > \varphi) = \max\{SD(< \alpha > \varphi), SD(< \beta > \varphi)\}$.
- $SD(< \alpha \cap \beta > \varphi) = \max\{SD(< \alpha > \varphi), SD(< \beta > \varphi)$.
- $SD([\alpha \cup \beta]\varphi) = \max\{SD([\alpha]\varphi), SD([\beta]\varphi)\}$.

- $SD([\alpha \cap \beta]\varphi) = \max\{SD([\alpha]\varphi), SD([\beta]\varphi)\}$.
- *If* $< \alpha' > \varphi$ *and* $[\beta']\varphi$ *are the diamond and box normal forms of* $< \alpha > \varphi$ *and* $[\beta]\varphi$ *then*

$$\begin{aligned} SD(< \alpha' > \varphi) &= SD(< \alpha > \varphi) \\ SD([\beta']\varphi) &= SD([\alpha]\varphi) \\ MD(< \alpha' > \varphi) &= MD(< \alpha > \varphi) \\ MD([\beta']\varphi) &= MD([\beta]\varphi). \end{aligned}$$

- *If* α *is a program free of* $*$*'s, it is in normal form, and* ψ_1 *and* ψ_2 *are the diamond stripped and box stripped normal forms of* $< \alpha > \varphi$ *and* $[\alpha]\varphi$ *then*

$$\begin{aligned} MD(< \alpha > \varphi) &= 1 + MD(\psi_1) \\ MD([\alpha]\varphi) &= 1 + MD(\psi_2). \end{aligned}$$

Similar relationships may not hold between the $*$*-depths, SD.*

- *If* $(\alpha; \beta^*; \gamma)$ *is in micronormal form where* α *is a composition of assignments and tests then*

$$\begin{aligned} SD(< \alpha; \beta^*; \gamma > \varphi) &> SD(< \alpha; \beta^i; \gamma > \varphi) \\ MD(< \alpha; \beta^*; \gamma > \varphi) &= MD(< \alpha; \beta^i; \gamma > \varphi) \end{aligned}$$

for every integer i*. This is true even when one or both of* α, γ *may be missing.*

- *Suppose that* α *is* $\cap_{i=1}^{n}(\alpha_i; \beta_i^*; \gamma_i)$ *which is in subnormal form where each* α_i *is a composition of tests and assignments; then*

$$\begin{aligned} SD([\alpha]\varphi) &> SD\left(\left[\bigcap_{i=1}^{n}(\alpha_i; \beta_i^{k_i}; \gamma_i)\right]\varphi\right) \\ MD([\alpha]\varphi) &= MD\left(\left[\bigcap_{i=1}^{n}(\alpha_i; \beta_i^{k_i}; \gamma_i)\right]\varphi\right) \end{aligned}$$

for any integers $k_1, \ldots, k_n$*. This is true even in the case where one or both of* α_i, γ_i *may be missing for some* i*.*

Proof. Most of these are proved by definition chasing or by directly applying the previous lemma. □

Now we construct the well ordering necessary to prove the MODEL EXISTENCE THEOREM.

Definition 10.2.7 (*Well Ordering* $\prec$). For each formula φ let $\#(\varphi)$ be the number of logical connectives of φ. Let $MD(\varphi)$ and $SD(\varphi)$ respectively be the modal and $*$-depths of φ. Let TRIPLE be defined as

$$\{(MD(\varphi), SD(\varphi), \#(\varphi)) : \varphi \text{ is a formula}\}.$$

Let $\prec$ be the lexicographic ordering on TRIPLE.

Notice that the lexicographical ordering on TRIPLE is a well ordering on the set of CCDL formulas. We use induction on this ordering to prove the model existence theorem.

10.3. Model existence theorem

Theorem 10.3.1 (*Model Existence Theorem*). *Suppose that* $(U, \leq)$ *is a consistency property. Then there is a Kripke frame* K *and an associated set of states* $S(K)$ *with a mapping* $\mathcal{F} : (U, \leq) \to S(K)$ *satisfying*

1. *If* $u_1 \leq u_2$ *then* $\mathcal{F}(u_1) \leq \mathcal{F}(u_2)$.
2. *Suppose that* $u \in U$ *and* $s \in S(K)$ *satisfies* $s > \mathcal{F}(u)$. *Then there is a* $u' \in U$ *satisfying* $u' > u$ *and* $\mathcal{F}(u_2) = s$.
3. *If* $T(\varphi) \in u$ *then* $\mathcal{F}(u) \vdash \varphi$ *and if* $F(\varphi) \in u$ *then* $\mathcal{F}(u) \nvdash \varphi$ *provided that all terms appearing in* φ *are defined in* $\mathcal{F}(u)$.

Proof. We first construct the Kripke frame K, define the set of partial states $S(k)$ over K, construct the mapping $\mathcal{F} : U \to S(K)$ and then show that our construction works.

1. Construction

1.1 Construction of the Kripke frame

Let $(U, \leq)$ be a consistency property of CCDL formulas over an enlarged set of individual constants $C \cup P$. We describe the construction of a Kripke frame

$$((K, \leq), \{(D(k), =_k, \{A_{i,k} : i \in \omega\}, \{g_{j,k} : j \in \omega\}) : k \in K\})$$

in the following steps. (We refer to these steps by their numbers at a later stage.)

1. Take $(K, \leq)$ to be $(U, \leq)$.
2. For each $u \in U$ let $P(u)$ consist of all constants (from both C and P) mentioned in terms of signed formulas of u and the set of individual constants. Take
$$D(u) = \bigcup_{u' \leq u} P(u').$$
3. Define a binary relation $=_u$ on $D(u)$ as $a =_u b$ if $T(a = b) \in u$.
4. For each predicate symbol A_i (of arity n) of the language and each $u \in U$ define
$$A_{i,u} \text{ as } \{(a_1, \ldots, a_n) : T(A_i(a_1, \ldots, a_n)) \in u \text{ for } (a_1, \ldots, a_n) \in D(u)^n\}.$$
5. For each n-ary function g_i of the language and each $u \in U$ define a partial function $g_{i,u}$ on $D(u)^n$ by $g_{i,u}(a_1, \ldots, a_n) = a$ for $a_1, \ldots, a_n, a \in D(u)$ provided that $T(g_i(a_1, \ldots, a_n) = a) \in u$.

Now in order to show that K is a Kripke model we need to show the following facts.

1. $D(u)$ is monotonic in u.
2. $A_{i,u}$ is monotonic in u for each i.
3. $g_{i,u}$ is single valued and graph $(g_{i,u})$ is monotonic in u for each i.
4. $=_u$ is reflexive, transitive, symmetric, and monotonic in u.

Proofs of these:

1. $D(u)$ is monotonic by construction (step 2).
2. By 1(a) of the consistency properties, if $T(A_i(a_1, \ldots, a_n)) \in u$ and $u' \geq u$ then $T(A_i(a_1, \ldots, a_n)) \in u'$ which gives monotonicity of $A_{i,u}$.

3. Suppose that $g_{i,u}(a_1, \ldots, a_n) =_u a$. Then $T(g_i(a_1, \ldots, a_n)) \in u$. By reasoning as in (2) we get the result for $u' \geq u$. The single valuedness of g_i is guaranteed by conditions 2(e) and 2(p) of the consistency properties.
4. Monotonicity, reflexivity, symmetry, and transitivity $=_u$ are consequences of consistency conditions 2(c) and 2(e).

1.2 Construction of partial states and forcing

For each $u \in U$ define a mapping $S_u : X \to D(u)$ as $S_u(x_i) = a_i \in D(u)$ where $a_i \in D(u)$ satisfies $T(x_i = a_i) \in u$. The existence and uniqueness of such an a_i is guaranteed by condition 2(p) of a consistency property. Notice that functions S_u map the set of program variables into states, thereby giving the collection of partial states $S(K)$. Now we extend S_u to all program terms that appear in u as in Definition 4.2.5. We also partially order $S(K)$ and define forcing of CCDL formulas by partial states as in Definitions 4.2.6 and 4.2.7.

1.3 Construction of the map $\mathcal{F} : (U, \leq) \to S(K)$

Define $\mathcal{F} : (U, \leq) \to S(K)$ as $\mathcal{F}(u) = S_u$.

2. Proving that our construction works

We need to prove that conditions (1), (2), and (3) of the statement of the model existence theorem are true.
(1) Suppose that $u_1 < u_2$. Then constructions of $D(u_1)$, $D(u_2)$, the partial states, and their ordering give $S(u_1) < S(u_2)$.
(2) Suppose that $u \in U$ and $s \in S(K)$ satisfy $s > \mathcal{F}(u)$. Then by the definition of $\leq$ on $S(K)$, s must be some $\mathcal{F}(u')$ for $u' \geq u$.
(3) This is proved by induction on the well ordering $\prec$ on CCDL formulas.
Inductive Assumption: For all formulas $\psi \prec \varphi$ and for all $u \in U$:

1. If $T(\psi) \in u$ then all terms in ψ exist in $\mathcal{F}(u)$ and

$$\mathcal{F}(u) \vdash \varphi.$$

2. If $F(\varphi) \in u$ then all terms in ψ exist in $\mathcal{F}(u)$ and

$$\mathcal{F}(u) \nvdash \varphi.$$

Base case of the inductive proof

Suppose that $MD(\varphi) = \#(\varphi) = 0$ and $SD(\varphi) = 0_0$. Then φ is either an atomic instance or an instance of equality.

Case 1. $T(\varphi) \in u$.
Suppose that $x_1, \ldots, x_n$ are all the state variables that appear in φ. Because of all instances of SV included in u (consistency conditions 2(p) and 2(f)) there are constants $c_1, \ldots, c_n \in C \cup P$ satisfying $T(x_i = c_i)$ for $i = 1, \ldots, n$. Now if φ is an atomic predicate then because all instances of axiom pred-6 are included in u (due to 2(p), 2(c), and 2(e) of the consistency conditions) we have that $T(\varphi(c_i/x_i)) \in u$. If φ is the equality predicate

then we have to use axiom fun-2 instead to get $T(\varphi(c_i/x_i)) \in u$. From steps 3, 4, and 5 of the construction of the Kripke frame K we get that $u \vdash \varphi(c_i/x_i)$. Now Lemma 4.2.8 gives $\mathcal{F}(u) \vdash \varphi$.

Case 2. $F(\varphi) \in u$.

Let $x_i, i = 1, \ldots, n$ be as in case 1. Then as in case 1 we get a sequence of constants c_i satisfying $T(c_i = x_i) \in u$. Now if $T(\varphi(c_i/x_i)) \in u$, because of condition 2(q) of the consistency properties, we get a contradiction of $T(\varphi(c_i/x_i)), F(\varphi(c_i/x_i)) \in u$. Therefore $T((\varphi(c_i/x_i)) \notin u$. Now by the construction of K and definition of forcing, we get $\mathcal{F} \nVdash \varphi(c_i/x_i)$ and hence $\mathcal{F} \nVdash \varphi$ by Lemma 4.2.8.

Inductive case of the inductive proof

Suppose that the inductive assumption is valid for all CCDL formulas ψ satisfying $\psi \prec \varphi$. We want to conclude that φ satisfies the inductive assumption. We have to go through an exhaustive case analysis depending upon the outermost logical connective of φ.

Case 1.T. φ is $\varphi_1 \wedge \varphi_2$ and $T(\varphi) \in u$.
Then $\varphi_i \prec \varphi$ for $i = 1, 2$. By 2(a) of the consistency properties we have that $T(\varphi_i) \in u$ for $i = 1, 2$. Then by the inductive assumption we get $\mathcal{F}(u) \vdash \varphi_i$ for $i = 1, 2$ giving $\mathcal{F}(u) \vdash \varphi_1 \wedge \varphi_2$.

Case 1.F. φ is $\varphi_1 \wedge \varphi_2$ and $F(\varphi) \in u$.
Then $\varphi_i \prec \varphi$ for $i = 1, 2$. By 2(a) of the consistency properties we have that $F(\varphi_i) \in u$ for either $i = 1$ or 2. Then by the inductive assumption we get $\mathcal{F}(u) \nVdash \varphi_i$ for either $i = 1$ or 2, giving $\mathcal{F}(u) \nVdash \varphi_1 \wedge \varphi_2$.

The cases when φ is $\varphi_1 \vee \varphi_2, \varphi_1 \rightarrow \varphi_2, \neg\varphi, \forall y\varphi_1(y), \exists y\varphi_1(y)$ are similar to the above case and consistency conditions 2(b) to 2(f) take care of them. We omit the repetition of these standard proofs. The remaining two cases deal with modal connectives.

Case 2.T. φ is $< \alpha > \psi$ and $T(\varphi) \in u$.
Let $< \alpha' > \psi$ be the diamond normal reduct of $< \alpha > \psi$. If $T(< \alpha > \psi) \in u$ then by 2(g) of the consistency properties we get $T(< \alpha' > \psi) \in u$.

Case 2.T.1. α' has $*$'s in it.
Let α' be $\cup_{i=1}^{n} \alpha_i$ where each α_i is of the form $\cap_{j=1}^{k_i}(\beta_{i,j}; \gamma_{i,j}^*; \theta_{i,j})$ where $\beta_{i,j}$ is a composition of tests and assignments. Notice that we include all degenerate cases where one or both of $\beta_{i,j}, \gamma_{i,j}$ may be missing for some n.

If $n > 1$ then by consistency condition 2(h) we have that

$$T(< \alpha_1 > \psi \vee < \alpha_2 > \psi \vee \cdots \vee < \alpha_n > \psi) \in u.$$

Hence by consistency condition 2(b), we get $T(< \alpha_i > \varphi_1) \in u$ for some i. Consistency condition 2(j) implies that

$$T(< \alpha_{i,j}; \beta_{i,j}^k; \gamma_{i,j} > \psi) \in u$$

for some integer k. Notice that

$$< \alpha_{i,j}; \beta_{i,j}^*; \gamma_{i,j} > \psi \prec < \alpha_{i,j}; \beta_{i,j}^k; \gamma_{i,j} > \psi.$$

Now the inductive assumption implies that

$$\mathcal{F}(u) \vdash < \alpha_{i,j}; \beta_{i,j}^{k}; \gamma_{i,j} > \psi.$$

Then the semantics of the $*$ operation implies that

$$\mathcal{F}(u) \vdash < \alpha_{i,j}; \beta_{i,j}^{*}; \gamma_{i,j} > \psi$$

giving

$$\mathcal{F}(u) \vdash < \alpha' > \psi$$

Equisatisfiability of $< \alpha > \psi$ and $< \alpha' > \psi$ implies that

$$\mathcal{F}(u) \vdash < \alpha > \psi.$$

Case 2.T.2. α' does not have any $*$'s.
Let ψ' be the modal stripped reduct of $< \alpha' > \psi$. Consistency condition 2(k) implies that $T(\psi') \in u$. Notice that ψ' has a strictly lesser modal depth, thereby making $\psi' \prec \varphi$. Now we apply the inductive assumption to ψ' and obtain

$$\mathcal{F}(u) \vdash \psi'.$$

The equisatisfiability of ψ', $< \alpha' > \psi$ and $< \alpha > \psi$ implies that

$$\mathcal{F}(u) \vdash < \alpha > \psi.$$

Case 2.F. φ is $< \alpha > \psi$ and $F(< \alpha > \psi) \in u$.
By 2(g) of the consistency properties we get $F(< \alpha' > \psi) \in u$ where $< \alpha' > \psi$ is the diamond normal form of $< \alpha > \psi$.

Case 2.F.1. α' does not have any $*$'s.
This proof parallels the one given for case 2.T.1. To sketch it out briefly, consistency condition 2(g) implies that $F(\psi') \in u$ where ψ' is the modal stripped reduct of $< \alpha' > \psi$. Now applying the inductive assumption to $F(\psi') \in u$ gives

$$\mathcal{F}(u) \nvdash \psi'.$$

Now the equisatisfiability of ψ', $< \alpha' > \psi$ and φ implies that

$$\mathcal{F}(u) \nvdash \varphi.$$

Case 2.F.2. α' has $*$'s.
This is the dual argument of case 2.T.1. We use the terminology of that case. Here we have

$$F(< \alpha' > \psi) \in u.$$

Consistency conditions 2(h) and 2(b) imply that

$$F(< \alpha_i > \psi) \in u \text{ for every } i \leq n.$$

Consistency condition 2(i) implies that

$$F(< \alpha_{i,j}; \beta_{i,j}^{*}; \gamma_{i,j} > \psi) \in u \text{ for every } i, j.$$

Consistency condition 2(j) implies that

$$F(< \alpha_{i,j}; \beta_{i,j}^m; \gamma_{i,j} > \psi) \in u \text{ for every integer } m \text{ for every integer } i, j.$$

Notice that

$$< \alpha_{i,j}; \beta_{i,j}^m; \gamma_{i,j} > \psi \prec< \alpha_{i,j}; \beta_{i,j}^*; \gamma_{i,j} > \psi$$

because the formula on the left has the same modal depth as but a lesser $*$-depth than the one on the right. Now the inductive assumption gives as

$$\mathcal{F}(u) \nvdash < \alpha_{i,j}; \beta_{i,j}^m; \gamma_{i,j} > \psi \text{ for every } m.$$

Now the semantics of $*$ gives that

$$\mathcal{F}(u) \nvdash < \alpha_{i,j}; \beta_{i,j}^*; \gamma_{i,j} > \psi.$$

The semantic equivalence of

$$\bigwedge_{j=1}^{n_i} < \alpha_{ij}; \beta_{ij}^*; \gamma_{ij} > \psi \text{ and } < \bigcap_{j=1}^{n_i} (\alpha_{ij}; \beta_{ij}^*; \gamma_{ij}) > \psi$$

implies that

$$\mathcal{F}(u) \nvdash < \alpha_i > \psi.$$

Now semantic equivalence of

$$< \alpha' > \psi \text{ and } \bigvee_{i=1}^{n} < \alpha_i > \psi$$

implies that

$$\mathcal{F}(u) \nvdash < \alpha_i > \psi.$$

Case 3.T. φ is $[\alpha]\psi$ and $T(\varphi) \in u$.
By consistency property 2(l) we get that $T([\alpha']\psi) \in u$ where $[\alpha']\psi$ is the box normal form of $[\alpha]\psi$.

Case 3.T.1. α' does not have any $*$'s.
Let ψ' be the modal stripped box reduct of $[\alpha']\psi$. Consistency condition 2(o) implies that $T(\psi') \in u$. Notice that

$$\psi' \prec [\alpha]\psi$$

because $MD(\psi') < MD([\alpha]\psi)$. The inductive assumption implies that

$$\mathcal{F}(u) \vdash \psi'.$$

Semantic equivalence of ψ' and $[\alpha]\psi$ gives

$$\mathcal{F}(u) \vdash [\alpha]\psi.$$

Case 3.T.2. α' has $*$'s in it.
Let α' be $\cup_{i=1}^{n}\alpha_i$ where α_i is $\cap_{j=1}^{n_i}(\alpha_{i,j};\beta_{i,j}^{*};\gamma_{i,j})$ where $\alpha_{i,j}$ is a composition of assignments and tests. We include all degenerate cases of α'. Consistency properties 2(m), 2(a) with $T([\alpha']\psi) \in u$ imply that $T([\alpha_i]\psi)$ for each $i \leq n$. Consistency property 2(n) implies that

$$T\left(\left[\bigcap_{j=1}^{n_i}(\alpha_{i,j};\beta_{i,j}^{k_{i,j}};\gamma_{i,j})\right]\psi\right) \in u$$

for any combination of integers $k_{i,j}$ satisfying $i \leq n$ and $j \leq n_i$. Notice that

$$\left[\bigcap_{j=1}^{n_i}(\alpha_{i,j};\beta_{i,j}^{k_{i,j}};\gamma_{i,j})\right]\psi \prec \varphi$$

because

$$SD\left(\left[\bigcap_{j=1}^{n_i}(\alpha_{i,j};\beta_{i,j}^{k_{i,j}};\gamma_{i,j})\right]\psi\right) < SD(\varphi).$$

Now by applying the inductive assumption to

$$T\left(\left[\bigcap_{j=i}^{n_i}(\alpha_{i,j};\beta_{i,j}^{k_{i,j}};\gamma_{i,j})\right]\psi\right) \in u$$

for each i and each integer $k_{i,j}$ we get

$$\mathcal{F}(u) \vdash \left[\bigcap_{j=1}^{n_i}(\alpha_{i,j};\beta_{i,j}^{k_{i,j}};\gamma_{i,j})\right]\psi.$$

Now the semantics of $*$ implies that

$$\mathcal{F}(u) \vdash \left[\bigcap_{j=1}^{n_i}(\alpha_{i,j};\beta_{i,j}^{*};\gamma_{i,j})\right]\psi.$$

The semantic equivalence of

$[\alpha']\psi$ and $\bigwedge_{i=1}^{n}([\alpha_i]\psi)$ imply that

$$\mathcal{F} \vdash [\alpha']\psi$$

implying that

$$\mathcal{F} \vdash \varphi.$$

Case 3.F. φ is $[\alpha]\psi$ and $F(\varphi) \in u$.
By consistency condition 2(l) we get that $F([\alpha']\psi) \in u$ where $[\alpha']\psi$ is the box normal form of $[\alpha]\psi$.

Case 3.F.1. α' does not have any $*$'s in it.
This proof similar to that of case 3.T.1. Briefly 2(o) implies that $F(\psi') \in u$ where ψ' is the modal stripped box reduct of $[\alpha']\psi$. The inductive assumption implies that $\mathcal{F}(u) \nVdash \psi'$ because $\psi' \prec [\alpha']\psi$. Equisatisfiability of ψ' and $[\alpha']\psi$ implies that

$$\mathcal{F}(u) \nVdash \varphi.$$

Case 3.F.2. α' has $*$'s in it.
We adopt the same terminology as in case 3.T.2 with the same possibilities of degenerate cases. Consistency properties 2(1) and 2(m) imply that

$$F([\alpha_i]\psi) \in u \text{ for some } i \leq n.$$

Consistency condition 2(n) implies that

$$F\left(\left[\bigcap_{j=1}^{n_i} \alpha_{i,j}; \beta_{i,j}^{k_{ij}}; \gamma_{i,j}\right]\psi\right) \in u'$$

for some $u' \geq u$ and some combination of integers $k_{i,j}$ for $j \leq n_i$. Notice that

$$\left[\bigcap_{j=1}^{n_i} (\alpha_{i,j}; \beta_{i,j}^{k_{i,j}}; \gamma_{i,j})\right]\psi \prec \varphi$$

because

$$SD\left(\left[\bigcap_{j=1}^{n_i} (\alpha_{i,j}; \beta_{i,j}^{k_{i,j}}; \gamma_{i,j})\right]\psi\right) < SD(\varphi).$$

The induction assumption implies that

$$\mathcal{F}(u') \nVdash \left[\bigcap_{j=i}^{n_i} (\alpha_{i,j}; \beta_{i,j}^{k_{i,j}}; \gamma_{i,j})\right]\psi.$$

The semantics of $*$ implies that

$$\mathcal{F}(u') \nVdash \left[\bigcap_{j=i}^{n_i} (\alpha_{i,j}; \beta_{i,j}^{*}; \gamma_{i,j})\right]\psi.$$

Equisatisfiability of $[\alpha']\psi$ and $\wedge_{i=1}^{n}([\alpha_i]\psi)$ implies that

$$\mathcal{F}(u') \nVdash [\alpha']\varphi.$$

The semantics of $\Box$ implies that

$$\mathcal{F}(u) \nVdash [\alpha']\psi$$

because $\mathcal{F}(u') \geq \mathcal{F}(u)$. Equisatisfiability of $[\alpha']\psi$ and φ implies that

$$\mathcal{F}(u) \nVdash \varphi. \quad \Box$$

11. Systematic tableaux

The tableau rules that were described earlier *may* be used, although they do not have to be used. In this section we describe the systematic tableau procedure that has the advantage of producing a proof, if there is one at all. As was done earlier, we build an ω^ω tree of signed formulas. To give a succinct presentation of all the different cases involved we extend Fitting's [5] uniform modal notation to CCDL tableaux.

11.1. Uniform modal notation extended

Recall that we used (ρ, σ, S) as notation for entries on the tableaux Ω. Here ρ and σ are finite sequences of integers and S is a set of signed sentences of CCDL over a language that has additional constants. The first coordinate of (ρ, σ, S), i.e. ρ, is the name of the node of the tableaux that we are constructing. The second coordinate σ is the name of the world in the Kripke frame that our model is attempting to create and the third coordinate S gives the set of formulas that σ is supposed to force (these appear with a T in front of them) and the set of formulas that σ is not suppose to force. (These appear with a F in front of them.) First we give the definition of a branch in our tableaux.

Definition 11.1.1 (*Branch*). We call a branch of the tableaux Ω a subset $\Omega' \subset \Omega$ satisfying:

- If $(\rho, \sigma, S), (\rho', \sigma', S') \in \Omega'$ then either $\rho \leq \rho'$ or $\rho' \leq \rho$.
- If $(\rho, \sigma, S) \in \Omega', (\rho', \sigma', S') \in \Omega$ and $\rho' \leq \rho$ then $(\rho, \sigma, S) \in \Omega'$.

Notice that this gives a branch in our tableaux and NOT a path in the Kripke frame. Next we provide some preliminary definitions.

Definition 11.1.2 (*Abandoned Branches*). Call a branch μ on a tableau abandoned if there is some CCDL formula φ and some ρ, σ satisfying $T(\varphi), F(\varphi) \in S$ with $(\rho, \sigma, S) \in \mu$. If μ is not abandoned then call it unabandoned, or an open branch.

Definition 11.1.3 (*Largest Entry of a World on a Branch*). Call (ρ, σ, S) the largest entry corresponding to the Kripke world σ on a (perhaps partially constructed) branch μ if

- $(\rho, \sigma, S) \in \mu$,
- $(\rho', \sigma, S') \notin \mu$ for any $\rho' \geq \rho$.

Definition 11.1.4 (*Branch Modification Above a World*). Suppose that μ is a branch of the tableaux Ω and $(\rho, \sigma, S) \in \mu$. In the construction of the systematic tableau what we mean by "Applying branch modification above σ on μ" is carrying out the following procedure.

Suppose that (ρ_i, σ_i, S_i) for $i = 1, \ldots, n$ are the largest entries corresponding the Kripke worlds σ_i which are the only ones that appear on the branch μ satisfying the conditions $(\rho', \sigma', S') \in \mu$ and $\sigma' \geq \sigma$. Then add $(\delta^\frown << 0^i >>, \sigma_i, S_i \cup \{T(\varphi) | T(\varphi) \in S\})$ to the tableaux where δ is the longest tableau node that currently exists on branch μ.

In constructing a tableau we need to apply tableau extension rules. The set of formulas that we attach to the newly created node depends upon the formula that we *analyze* for the current node. An inspection of our tableau extension rules reveals that we have to extend a branch with three kinds of extensions. They are unary, binary, or infinitary extensions.

Accordingly we categorize formulas as unary, binary, or infinitary. In the case for each of these extensions we have to determine the set of formulas that are to be attached to the new node or nodes. We also have to attach a name of a Kripke world to the new node on the tableau. For some of these we attach the name of the current Kripke world and in other cases we have to attach a name that has not been used on the current branch. Accordingly we have to again categorize a formula as *Present* or *Future* depending on the type of the label of the Kripke world to be attached. The set of formulas to be attached depends upon both the *unary:binary:infinitary* categorization and the *Present:Future* categorization of the formula under analysis.

11.1.1. Unary, binary, and infinitary formulas

In the case of unary formulas we need to specify what components are to be included in the new node. We call this component ψ_0 where we call the formula ψ. In the case of binary formulas we need two components which we call ψ_0 and ψ_1. For infinitary formulas we need a component for each integer i which we call ψ_i. We now present these in the following tables. In describing the systematic tableau construction we have 31 cases to consider and for this reason we write an integer called the *stage* along the formula. In the presentation of tableau rules this number was used to itemize the different cases.

Unary formulas

1. Propositional connectives

Stage	ψ	ψ_0
1	$T(\varphi \wedge \psi)$	$T(\varphi), T(\psi)$
4	$F(\varphi \vee \psi)$	$F(\varphi), F(\psi)$
6	$F(\varphi \rightarrow \psi)$	$T(\varphi), F(\psi)$
7	$T(\neg\varphi)$	$F(\varphi)$
8	$F(\neg\varphi)$	$T(\varphi)$

2. Modal formulas with $\Box$

Stage	ψ	ψ_0
9	$T([\alpha]\varphi)$	$T([\alpha']\varphi)$[100]
10	$F([\alpha]\varphi)$	$F([\alpha']\varphi)$[100]
11	$T([\alpha \cup \beta]\varphi)$	$T([\alpha]\varphi \wedge [\beta]\varphi)$
12	$F([\alpha \cup \beta]\varphi)$	$F([\alpha]\varphi \wedge [\beta]\varphi)$
14	$T([\alpha]\varphi)$	$T(\psi)$[101]
13	$T([\alpha]\varphi)$	$\{T(\cap_{i=1}^{n}(\alpha_i; \beta_i^{k_i}; \gamma_i)) : k_i \in \omega\}$[102]
16	$F([\alpha]\varphi)$	$F(\psi)$[101]

[100] $[\alpha']\varphi$ is the box normal reduct of $[\alpha]\varphi$.

[101] $[\alpha]\varphi$ is in subnormal form where α is free of $*$'s and ψ is the modal stripped reduct of $[\alpha]\varphi$.

[102] Here α is $\cap_{i=1}^{n}(\alpha_i; \beta_i^*; \gamma_i)$ in subnormal form.

3. Modal formulas with $\Diamond$

Stage	ψ	ψ_0
17	$T(<\alpha>\varphi)$	$T(<\alpha'>\varphi)$[201]
18	$F(<\alpha>\varphi)$	$F(<\alpha'>\varphi)$[201]
19	$T(<\alpha\cup\beta>\varphi)$	$T(<\alpha>\varphi\vee<\beta>\varphi)$
20	$F(<\alpha\cup\beta>\varphi)$	$F(<\alpha>\varphi\vee<\beta>\varphi)$
21	$T(<\alpha\cap\beta>\varphi)$	$T(<\alpha>\varphi\wedge<\beta>\varphi)$
22	$F(<\alpha\cap\beta>\varphi)$	$F(<\alpha>\varphi\wedge<\beta>\varphi)$
24	$T(<\alpha>\varphi)$	$T(\psi)$[202]
26	$F(<\alpha>\varphi)$	$F(\psi)$[202]
25	$F(<\alpha>\varphi)$	$\{F(<\alpha;\beta^i;\gamma>\varphi) : i\in\omega\}$[203]

4. Quantifiers

Stage	ψ	ψ_0
27	$T(\exists y\varphi(y))$	$T(a)$[300], $\{F(\varphi(a) : F(\exists y\varphi(y))\in S\}$
28	$F(\exists y\varphi(y))$	$\{F(\varphi(e))$: e appears in signed formulas of $S\}$
29	$T(\forall y\varphi(y))$	$\{T(\varphi(e))$: e appears in signed formulas of $S\}$
30	$F(\forall y\varphi(y))$	$F(\varphi(a))$[300]

Binary formulas

Stage	ψ	ψ_0	ψ_1
3	$T(\varphi\vee\psi)$	$T(\varphi)$	$T(\psi)$
2	$F(\varphi\wedge\psi)$	$F(\varphi)$	$F(\psi)$
5	$T(\varphi\to\psi)$	$F(\varphi)$	$T(\psi)$

Infinitary formulas

Stage	ψ	ψ_i
15	$F([\alpha]\varphi)$	$T([\cap_{j=1}^n\alpha_j;\beta_j^{k_j};\gamma_j]\varphi)$[500]
23	$T(<\alpha>\varphi)$	$T(<\beta;\gamma^i;\delta>\varphi)$[501]

201 $<\alpha'>\varphi$ is the diamond normal reduct of $<\alpha>\varphi$.

202 Here α, $*$-free, is in diamond normal form and ψ is the modal stripped reduct of $<\alpha>\varphi$.

203 $\alpha=(\alpha;\beta^*;\gamma)$ is in diamond normal form.

300 a is a new constant on this branch.

500 Here α in subnormal form is $\cap_{j=1}^n(\alpha_j;\beta_j^*;\gamma_j)$ and i is $<< j_1,\ldots,j_n >>$.

501 α in micronormal form is $(\beta;\gamma^*;\delta)$.

11.1.2. Future and Present formulas

The following are *Future* formulas: $F(\varphi \rightarrow \psi)$, $F(\neg\varphi)$, $F(\forall x\varphi(x))$, and $F([\alpha]\varphi)$ where α is in subnormal form and has $*$'s. The others are *Present* formulas.

11.2. Systematic tableau procedure

We build the systematic tableau of the deduction $\{F\varphi\} \cup \{T(\psi_i) : i \in \omega\}$ in stages. We assume the existence of a countable number of new constants $\{d_i : i \in \omega\}$.

Convention. We assume that we have a Gödel numbering schema in which a formula is numbered *compatible* with the stage number assigned in the tables given above in the sense that if it is a unary formula with stage number i then its *Gödel* number is (31 mod i). We further assume that every formula appears infinitely many times on this list. The same relations hold for binary and infinitary formulas. By the least formula we mean the formula with least Gödel number with respect to our numbering schema. We use the notation S^T for $\{T(\varphi)|T(\varphi) \in S\}$.

Definition 11.2.1 (*Extending a Branch*). Suppose that μ is a partially completed branch on a tableau and $(\rho, \sigma, S) \in \mu$ where ρ is the largest tableau node on μ (i.e. $(\rho', \sigma', S) \notin \mu$ for any $\rho' > \rho$). Then we refer to $\rho^\frown << i >>$ as μ_i.

The systematic tableau algorithm uses the following two subprocedures.

Definition 11.2.2 (*Generic Branch Extension Algorithm*). Suppose that (ρ', σ', S') is on the tableaux, i is an integer, and ψ, ψ_i are formulas defined according to the environment in which this procedure is used. Then we define (ρ_i, σ_i, S_i) according to the following schema.

- ρ_i is μ_i.
- If ψ is a *Present* formula then σ_i is σ'; else let σ'' be the name of the Kripke world that is the least one not yet on branch μ.
- If ψ is a *Present* formula let S_i be S, ψ_i; else let S_i be S^T, ψ_i.

Definition 11.2.3 (*Generic Branch Modification Algorithm*). For each i if ψ_i is a formula signed with T and S_i is S, ψ_i then apply branch modification rule above σ on branches including μ_i. Here i and σ are the parameters of the procedure.

Systematic tableau algorithm

The systematic tableau is created in ω many stages according to the following schema.

Stage 0: Create the root of the tableau by including $(<< 0 >>, << 0 >>, \{F(\varphi)\} \cup \{T(\psi_i)|i \in \omega\})$ in tableaux Ω.

Stage n; for $n > 0$**:**

We have to consider 31 cases; hence we consider stage numbers modulo 31. Let k be (n mod 31).

(1) $k = 0$; Adding axioms

If k = 0 then for each unabandoned branch μ on the tableaux let $(\rho', \sigma', S') \in \mu$ be the one with the least σ satisfying $T(AX_m) \notin S$ for the least such m. Then extend each branch μ by adding $(\mu_0, \sigma', S' \cup \{T(AX_m)\})$. Apply the branch modification rule above σ'.

(2) $k \neq 0$

Case 1: Unary formula. Suppose that stage number k belongs to a unary formula. For each open branch μ of the tableaux let ψ be the formula with the least *Gödel* number that satisfies the following conditions: the stage number of ψ is k, $\psi \in S$, and $\psi_0 \notin S$ for some $(\rho, \sigma, S) \in \mu$. Among the (ρ, σ, S) that satisfy these conditions let (ρ', σ', S') be the one with the least σ. Then:

1. Extend each branch μ by adding (ρ_0, σ_0, S_0) where ρ_0, σ_0, S_0 is found according to the generic branch extension algorithm for $i = 0$ and ψ.
2. Apply the generic branch modification algorithm for $i = 0$ and σ.

Case 2: Binary formula. Suppose that stage number k belongs to a binary formula. For each open branch μ of the tableaux let ψ be the formula with the least Gödel number that satisfies the following conditions: the stage number of ψ is k, $\psi \in S$, and $\{\psi_0, \psi_1\} \not\subseteq S$ for some $(\rho, \sigma, S) \in \mu$. Among the (ρ, σ, S) that satisfy these conditions let (ρ', σ', S') be the one with the least σ. Then:

1. Extend each branch μ into two branches by adding (ρ_i, σ_i, S_i) for $i = 1, 2$ where ρ_i, σ_i, S_i are found according to the generic branch extension algorithm for $i = 0, 1$.
2. Apply the generic branch modification algorithm for $i = 0, 1$ and σ.

Case 3: Infinitary formula. Suppose that stage number k belongs to an infinitary formula. For each open branch μ of the tableaux let ψ be the formula with the least Gödel number that satisfies the following conditions: the stage number of ψ is k, $\psi \in S$, and $\{\psi_i | i \in \omega\} \not\subseteq S$ for some $(\rho, \sigma, S) \in \mu$. Among the (ρ, σ, S) that satisfy these conditions let (ρ', σ', S') be the one with the least σ. Then:

1. Extend each branch μ into ω many branches by adding (ρ_i, σ_i, S_i) for $i \in \omega$ where ρ_i, σ_i, S_i are found by applying the generic branch extension algorithm for σ_i for each $i \in \omega$.
2. Apply the generic branch modification algorithm for $i \in \omega$. □

12. Completeness of CCDL

In this section we show that CCDL is complete with respect to our tableau procedure. We do so by proving that an open branch of a systematic tableau gives a consistency property.

12.1. Consistency from an open branch

This subsection constructs a consistency property out of an open branch of a tableau.

Definition 12.1.1. Let β be a branch of a systematic tableau. Define a partially ordered collection of signed formulas $C(\beta) = (U, \leq)$ as follows.

- For every name of a Kripke world σ that appears on β with a set of signed sentences (i.e. there are ρ, S with $(\rho, \sigma, S) \in \beta$) let $U_\sigma = \cup\{S : (\rho, \sigma, S) \in \beta \text{ for some } \rho, S\}$.

- Let $U = \{U_\sigma : \sigma$ be the name of a Kripke world appearing on $\beta\}$.
- Define a partial ordering $\leq$ on U as $U_\sigma \leq U_{\sigma'}$ if σ is an initial segment of σ'.

In order to show that every open branch gives a consistency property we need to prove the following technical facts.

12.2. Some preliminary facts

Lemma 12.2.1. *Suppose that β is a partially constructed branch on a systematic tableau that is in the process of being constructed and let $(\rho_1, \sigma_1, S_1), (\rho_2, \sigma_2, S_2) \in \beta$ be the two largest entries corresponding to Kripke worlds σ_1, σ_2 at this time. Assume further that*

- *$\sigma \leq \sigma_1 \leq \sigma_2$ and $S_1^T \leq S_2^T$;*
- *we apply the branch modification rule above σ and (ρ_i, σ_i, S_i) becomes $(\rho_i', \sigma_i', S_i')$ for $i = 1, 2$.*

Then $(S_1')^T \subseteq (S_2')^T$.

Proof. By construction $S_i' = S_i \cup S^T$ for $i = 1, 2$. Hence $(S_1')^T \subseteq (S_2')^T$. □

Lemma 12.2.2. *Suppose that $(\rho, \sigma, S) \in \beta$ is the largest entry corresponding to the Kripke world σ on a partially completed branch β and we extend β by applying the branch extension rule for the* Future *formulas $F(\varphi \to \psi), F(\neg\varphi), F(\forall y \varphi(y))$, or $F([\alpha]\varphi)$.[12] (Notice that these are the steps at which it becomes necessary to append a new Kripke world.) Let the new node that we add be called $(\rho'.\sigma', S')$ where $\sigma' \geq \sigma$ and $\rho \geq \rho'$. Then $S'^T \supseteq S^T$.*

Proof. By construction we get S' by adding to S^T either $\{T(\varphi), F(\psi)\}, \{T(\varphi)\}$, $\{F(\varphi(e))\}$, or $\{F([\cap_{i=1}^n(\alpha_i; \beta_i; \gamma_i)]\}$ depending upon the specific branch extension rule applied. In all cases we get $S'^T \supseteq S^T$. □

Lemma 12.2.3. *Suppose that $(\rho, \sigma, S), (\rho', \sigma', S')$ are the largest entries corresponding to σ and σ' on the partially completed branch β of a systematic tableau. Suppose also that $\sigma < \sigma'$ and $S^T \subseteq S'^T$. Now suppose that we apply a tableau extension rule for a* Present *formula (i.e. one that does not require the addition of a new Kripke world above σ). Then we apply the branch modification above σ (because the extension rule requires us to do so). Let $(\rho_1, \sigma, S_1), (\rho_1', \sigma', S_1')$ be the new largest entries corresponding to Kripke worlds σ, σ'. Then $S_1^T \subseteq S_1'^T$.*

Proof. An inspection of the effect of applying branch extension followed by branch modification above σ will show this. □

Lemma 12.2.4. *Suppose that $(\rho, \sigma, S), (\rho', \sigma', S')$ are the largest entries corresponding to σ and σ' on the partially completed branch β of a systematic tableau. Suppose also that $\sigma < \sigma'$ and $S^T \subseteq S'^T$. Now suppose that we apply a tableau extension rule that requires the addition of a new Kripke world above σ and then we apply the branch*

[12] Here α is subnormal.

modification above σ' (because the branch extension rule requires us to do so). Let (ρ_1, σ, S_1), (ρ_1', σ', S_1') be the new largest entries corresponding to the Kripke worlds σ and σ'. Then $S_1^T \subseteq S_1'^T$.

Proof. Same as for Lemma 12.2.3. □

Lemma 12.2.5. *Suppose that (ρ, σ, S), (ρ', σ', S') are the largest entries corresponding to σ, σ' on the partially completed branch β of a systematic tableau. Suppose also that the following hold:*

- *$\sigma < \sigma'$ and $S^T \subseteq S'^T$.*
- *$\sigma < \sigma'$.*
- *There is no Kripke world σ'' satisfying $\sigma < \sigma'' < \sigma'$ on β.*

Then $S'^T \supseteq S^T$.

Proof. Suppose that the first time σ' was introduced onto branch β we had (ρ_1, σ, S_1) and (ρ_2, σ', S_2). Then by Lemma 12.2.2 we get $S_1^T \subseteq S_2^T$. Now (ρ, σ, S) and (ρ', σ', S') are obtained by applying one of the following procedures one or many times.

- Apply some branch extension rule followed by the branch modification rule above σ to some $(\delta, \sigma, R) \in \beta$ with $\delta > \rho_1$.
- Apply the branch modification rule above σ'' for some $\sigma'' < \sigma$.
- Apply some branch extension rule followed by the branch modification rule to some $(\delta, \sigma', R) \in \beta$ with $\delta > \rho_2$.

The first case is taken care of by Lemma 12.2.3, the second by Lemma 12.2.1, and the third by Lemma 12.2.4. □

Lemma 12.2.6. *Suppose that (ρ, σ, S), (ρ', σ', S') are the largest entries corresponding to σ and σ' on the partially completed branch β of a systematic tableau satisfying $\sigma < \sigma'$. Then $S'^T \supseteq S^T$.*

Proof. There are only finitely many names for Kripke worlds σ_i with $\sigma_1 < \sigma_2 < \cdots < \sigma_n < \sigma'$ on branch β where (ρ_i, σ_i, S_i) is the largest entry corresponding to the Kripke world σ_i. Now by Lemma 12.2.5 we get $S^T \subset S_1^T \subseteq S_2^T \subseteq \cdots \subseteq S_n^T \subseteq S_{\sigma'}^T$. □

12.3. Consistency of open branches

In this section we show that for each open branch β, $C(\beta)$ gives a consistency property. Technically we are required to show that all the conditions of a consistency property are satisfied by $C(\beta)$. Notice that, from the perspective of uniform modal notation, these conditions and the tableau rules themselves can be categorized into three classes below, although the last class distinction is not very important, from the point of view of uniform modal notation and the structure of the tableau *tree*.

- Dealing with unary, binary, or infinitary formulas.
- Dealing with *Present* or *Future* formulas.
- Dealing with decoding of modal or non-modal connectives.

Accordingly, in our proof we provide an example from each of the relevant categories. In addition to these consistency conditions, 1(a) and 1(b) which read as

1(a) $S' \geq S$ and $T(\varphi) \in S$ imply $T(\varphi) \in S'$,
1(b) $T(\varphi), F(\varphi) \notin S$ for any S

do not belong in any of the above categories. We provide the proofs of these at the beginning.

Theorem 12.3.1. *Let β be an open branch of a systematic tableau. Then $C(\beta)$ is a consistency property.*

Proof. We have to show that $C(\beta)$ satisfies all requirements of a consistency property. Our numbering is the same as in the definition of a consistency property.
Consistency conditions 1(a) *To show:* If $U_\sigma, U_{\sigma'} \in U, U_\sigma < U_{\sigma'}$ and $T(\varphi) \in U_\sigma$ then $T(\varphi) \in U_{\sigma'}$.

Consider any stage t of the construction of the systematic tableau having the following properties.

- Kripke worlds σ, σ' appear on β; i.e., $(\rho, \sigma, S), (\rho', \sigma', S') \in \beta$ for some ρ, ρ', S, S'.
- $T(\varphi) \in S''$ for some $(\rho'', \sigma, S'') \in \beta$ at stage t.

Let $(\rho_1, \sigma, S_1), (\rho_2, \sigma', S_2)$ be the largest entries corresponding to Kripke worlds σ, σ' on β. By Lemma 12.2.6 we get $S^T \subseteq S_1^T \subseteq S_2^T$. Hence $T(\varphi) \in U_{\sigma'}$.
Consistency conditions 1(b) *To show:* For no $U_\sigma \in U$ do we have $T(\varphi), F(\varphi) \in U_\sigma$ for some sentence φ.

Suppose for a contradiction that this is not the case. Then there are tableau nodes ρ_1, ρ_2 with $T(\varphi) \in S_1$, $F(\varphi) \in S_2$ and $(\rho_i, \sigma, S_i) \in \beta$. Without loss of generality we assume that $\rho_1 < \rho_2$. Notice that by construction $S_1 \subset S_2$. Then we have $T(\varphi), F(\varphi) \in S_2$, contradicting the openness of branch β.

Unary Present non-modal formulas. The consistency conditions of the first parts of 2(a), 2(d), 2(e) and second parts of 2(b), 2(f), 2(p) belong to this category. We prove the first part of 2(a).
To show that: If $T(\psi \wedge \theta) \in U_\sigma$ then $T(\psi), T(\theta) \in U_\sigma$.

Suppose that $T(\psi \wedge \theta) \in S$ where $(\rho, \sigma, S) \in \beta$ and $T(\psi), T(\theta) \notin S$. At some stage of the construction after attaching (ρ, σ, S) on the tableaux, $T(\psi \wedge \theta)$ becomes the formula with the least Gödel number that satisfies $T(\varphi \wedge \varphi') \in S$ and $T(\varphi), T(\varphi') \notin S$ on partially completed β. Then by the branch extension rule at the next appropriate stage we attach $(\rho', \sigma, S \cup \{T(\psi), T(\theta)\})$ to β and apply branch modification above σ on β. Say that (ρ'', σ, S'') is the largest entry corresponding to the Kripke world σ on β right after this step has been completed. Then $T(\psi), T(\theta) \in S''$, and hence $T(\psi), T(\theta) \in U_\sigma$.

Binary Present non-modal formulas. The consistency conditions of the second parts of 2(a), 2(c) and first part of 2(b) belong to this category. We provide a proof of the second part of 2(a)
To show that: If $F(\psi \wedge \theta) \in U_\sigma$ then either $F(\psi)$ or $F(\theta) \in U_\sigma$.

Suppose that $F(\psi \wedge \theta) \in S$ where $(\rho, \sigma, S) \in \beta$ and $F(\psi), F(\theta) \notin S$. At some stage of the construction after attaching (ρ, σ, S) on the tableaux, $F(\psi \wedge \theta)$ becomes the formula with the least Gödel number that satisfies $F(\varphi \wedge \varphi') \in S$ and $F(\varphi), F(\varphi') \notin S$ on partially completed β. Then by the branch extension rule at the next stage $(31k+4)$ we attach $(\rho_1, \sigma, S \cup \{F(\psi)\})$, $(\rho_2, \sigma, S \cup \{F(\theta)\})$ to β and apply branch modification above σ on β. Here ρ_i is β_i' where β' is partially completed β just before applying the branch extension rule. Notice that one of ρ_1, ρ_2 belongs to β. Say that (ρ_2, σ, S'') is the largest entry corresponding to the Kripke world σ right after the above-mentioned branch extension. Then $F(\theta) \in S''$, and hence $F(\theta) \in U_\sigma$.

Unary Future non-modal formulas. The second parts of consistency conditions 2(c), 2(d), and 2(e) belong to this category. We provide a proof of the second part of 2(c).
To show: If $F(\psi \to \theta) \in U_\sigma$ then $T(\psi), F(\theta) \in U_{\sigma'}$ for some $\sigma' \geq \sigma$.

Let $(\rho, \sigma, S) \in \beta$ be an entry on branch β with $F(\psi \to \theta) \in S$ and $T(\psi), F(\theta) \notin S'$ for any S' satisfying $(\rho', \sigma', S') \in \beta$ for some $\sigma' \geq \sigma$ and some ρ'. Then there is a stage $(31k+6)$ after attaching (ρ, σ, S) to β at which $F(\psi \to \theta)$ becomes the formula with the least Gödel number satisfying the stated conditions. Let $(\rho', \sigma, S') \in \beta$ be the largest entry corresponding to the Kripke world σ at this stage of the construction. At this stage, by the tableau extension rule $(31k+6)$ we attach $(\beta_1', \sigma'', S' \cup \{T(\psi), F(\theta)\})$ to the partially completed branch β' of β for some $\sigma'' \geq \sigma$ fresh on β'. Then $T(\psi), F(\theta) \in U_{\sigma''}$.

Unary Present modal formulas. Consistency conditions 2(g) through 2(i), 2(l), 2(m), 2(o), 2(p), the second part of 2(j) and the first part of 2(n) belong to this category. We prove the first part of 2(g).
To show: If $T(< \alpha > \varphi) \in U_\sigma$ then $T(< \alpha' > \varphi) \in U_\sigma$ where $< \alpha' > \varphi$ is the diamond normal reduct of $< \alpha > \varphi$.

Suppose that at some stage t of the construction we have $(\rho, \sigma, S) \in \beta$ with $< \alpha > \varphi \in S$ and $< \alpha' > \varphi \notin S$. Then there is a stage $(31k+17) > t$ at which $< \alpha > \varphi$ becomes the formula with the least Gödel number that satisfies the property that $< \alpha > \varphi \in S_1$ and $< \alpha' > \varphi \notin S_1$ where (ρ', σ', S_1) is the largest entry corresponding to the Kripke world σ at that stage. Then, by the branch extension rule $31k+17$ we add $(\rho', \sigma, S \cup \{T(< \alpha' > \varphi\})$ to β. Now we have $< \alpha' > \varphi \in U_\sigma$.

Infinitary Present modal formulas. Consistency condition 2(j) belongs to this category.
To show: If $T(< \alpha; \delta^*; \gamma > \varphi) \in U_\sigma$ then $T(< \alpha; \delta^i; \gamma > \varphi) \in U_\sigma$ for some integer i.

Suppose that $T(< \alpha; \delta^*; \gamma > \varphi) \in S$ and $T(< \alpha; \delta^i; \gamma > \varphi) \notin S$ for any integer i where $(\rho, \sigma, S) \in \delta$ in some stage of the construction of the tableaux. Then there is a stage $31k+23$ at which (ρ, σ, S) becomes the node of the partially completed tableau that has the largest Kripke world corresponding to the formulas $< \alpha; \delta^*; \gamma > \varphi$ that have the least Gödel number among formulas satisfying $T(< \alpha; \delta^*; \gamma > \varphi) \in S$ and $T(< \alpha; \delta^i; \gamma > \varphi) \notin S$. At that stage we add $(\delta_i', \sigma, S \cup \{T(< \alpha; \delta^i; \gamma > \varphi)\})$ for each integer i to the tableaux, thereby creating an ω branching. Now the branch β contains one of these i, say the one indexed by j. Then $T(< \alpha; \delta^j; \gamma > \varphi) \in U_\sigma$. The degenerate cases of missing α, γ or both are handled in the same way.

Infinitary Future modal formulas. The second part of consistency condition 2(n) belongs to this category.

To show: If $F([\cap_{i=1}^{n}(\delta_i; \gamma_i^*; \theta_i)]\varphi) \in U_\sigma$ then $F([\cap_{i=1}^{n}(\delta_i; \gamma_i^{j_i}; \theta_i)]\varphi) \in U_{\sigma'}$ for some combinations of integers $j_1, \ldots, j_n$. We include the degenerate cases of one or both of δ_i, θ_i missing. Here δ's are compositions of assignments and tests.

Suppose that $F([\cap_{i=1}n(\delta_i; \gamma_i^*; \theta_i)]\varphi) \in U_\sigma$; then there is some stage t of the construction of the systematic tableaux at which $F([\cap_{i=1}^{n}(\delta_i; \gamma_i^*; \theta_i)]\varphi) \in S$ for some $(\rho, \sigma, S) \in \beta$ and there are no $\sigma' \geq \sigma$ and $(\rho', \sigma', S') \in \beta$ that satisfy $F([\cap_{i=1}^{n}(\delta_i; \gamma_i^{j_i}; \theta_i)]\varphi) \in S'$ at stage t. Call this condition the *prerequisite*. Then there comes a stage $31k + 15 > t$ at which $F([\cap_{i=1}^{n}(\beta_i; \gamma_i^*; \theta_i)]\varphi) \in S$ is the formula with the least Gödel number that satisfies the prerequisite. Let $(\rho', \sigma, S') \in \beta$ be the largest entry for the Kripke world σ on partially completed β (call it β') that satisfy the prerequisite. At this stage we apply branch extension rule $31k + 15$ and create an ω branching in the tableaux by adding $(\beta_i', \sigma', S^T \cup \{F([\cap_{i=1}^{n}(\beta_i; \gamma_i^{j_i}; \theta_i)]\varphi)\})$ for each i and for some $\sigma' > \sigma$ new on β'. Now the branch β contains one of these extensions, say β_i'. Then we have that $F([\cap_{i=1}^{n}(\delta_i; \gamma_i^{j_i}; \theta_i)]\varphi) \in U_{\sigma'}$ for some integers $j_1, \ldots, j_n$. □

12.4. Completeness of systematic tableaux

Theorem 12.4.1 (*Completeness of CCDL Tableaux*). *CCDL is complete with respect to our tableau procedure.*

Proof. Suppose that $\psi, \{\varphi_i : i \in \omega\}$ are CCDL sentences and that ψ is not tableau deducible from $\{\varphi_i : i \in \omega\}$. Then there is an open branch of the systematic tableaux with $(<< 0 >>, << 0 >>, \{F(\psi)\} \cup \{T(\varphi_i) : i \in \omega\})$ as the root node. Then Theorem 12.3.1 implies that there is a consistency property $(U, \leq)$ satisfying $\{F(\psi)\} \cup \{T(\varphi_i) : i \in \omega\} \subseteq v \in U$. The model existence theorem now says that there is a Kripke frame K and an associated set of states S over K and a state $s \in S$ with $s \vdash \varphi_i$ for each i but $s \nvdash \psi$. This concludes the proof of the completeness of the tableau decision procedure. □

13. Conclusions and future work

In this paper we have presented a constructive version of concurrent dynamic logic (called CCDL) that enables one to reason about statements that are true under partial information about the states of an underlying concurrent transition system.

The notion of *state* depends upon both the transition system and the *observables* that one uses to describe the state. The standard practice is to use variables to denote these observables. In any logical framework that describes programs and predicate logic one encounters two kinds of variables: one for use in assignments the other for use in quantification. Traditionally dynamic logic has been developed by merging these two kinds of variables into a single category. In this paper we adopt the opposite approach for the following reasons.

- Separation of variables makes the underlying transition system more apparent.
- It is not likely that a programmer envisions using names of memory locations (i.e. program variables) as variables for the quantificational logic.

- The observables of a programmer are hardly the *satisfaction sequences* in the sense of Traski.
- Once a distinction has been made between the two different usage of *variables* it is not difficult to merge them later on (if need be).

Despite our differing concept of state, that can be easily reconciled with the conventional notion of state in dynamic logic, we believe that the present work does the following.

1. Provides a clear definition of partial states and how a transition system may be developed on partial states.
2. Provides a more intuitive definition of $[\alpha]\varphi$ so that $\Box$ behaves more like a universal quantifier over trees in a concurrent language with constructs such as *parbegin* and *parend*.
3. Proves that every Peleg transition diagram has a corresponding *normal* description and provides the necessary axioms for deriving the *normal form theorems* in the framework of dynamic logic.
4. Supplies a complete tableau procedure.

The normal form theorems proved here carry over to the classical setting easily (irrespective of the distinction between the two kinds of variables). We are working on incorporating communication into concurrent dynamic logic (see [24]) and the corresponding process logic. We have noticed that multimedia systems can be modelled in the latter type of logic, although they are currently being described mainly in terms of Petri nets. Finally we remark that CCDL can be implemented on systems such as KIV (see [10]).

Acknowledgements

Anil Nerode was supported by NSF contract DMS-8902797 and ARO contract DAA 29-85-C108. Thanks go to J.N. Crossley and J.B. Remmel for many suggestions. The second author thanks the Army High Performance Research Center at the University of Minnesota for providing support.

References

[1] M. Bozic, K. Dosen, Models for normal intuitionistic logics, Studia Logica 43 (1984) 217–245.
[2] R. Burstall, Program proving as hand simulation with a little induction, Information Processing 74 (1974) 308–312.
[3] A. Chandra, D. Kozen, L. Stockmeyer, Alternation, Journal of the Association for Computing Machinery 28 (1981).
[4] K. Dosen, Models for strongly normal intuitionistic modal logics, Studia Logica 1 (1985).
[5] M.C. Fitting, Proof Methods for Modal and Intuitionistic Logic, Reidel, 1983.
[6] R. Floyd, Assigning meaning to programs, Proceedings of Symposia in Applied Mathematics 19 (1967) 19–32.
[7] D.M. Gabbay, Guenthner, Handbook of Philosophical Logic I, II, III, Riedel, 1983–1985.
[8] R.I. Goldblatt, Logic of Time and Computation, CSLI Lecture Notes, vol. 7, 1987.
[9] R.I. Goldblatt, Parallel Action: Concurrent dynamic logic with independent modalities. Technical Report 91-73, Victoria University of Wellington, 1991.

[10] R. Hähnle, M. Heisel, W. Reif, W. Stephan, An interactive verification system based on dynamic logic, in: J. Siekmann (Ed.), Proc. 8th Inter. Conf. on Automated Deduction, Lecture Notes in Computer Science, vol. 230, 1986, pp. 306–315.
[11] M. Heisel, W. Reif, W. Stephan, A functional language to construct proofs, Interner Bericht 1/86, Facultät für Informatik, Universität Karlsruhe, 1986.
[12] M. Heisel, W. Reif, W. Stephan, Program verification by symbolic execution and induction, in: K. Morik (Ed.), Proc. 11th German Workshop on Artificial Intelligence, Informatik Fachberichte, vol. 152, Springer-Verlag, 1987.
[13] M. Heisel, W. Reif, W. Stephan, Program verification using dynamic logic, in: E. Börger, H.K. Büning, M.M. Richter (Eds.), Workshop on Computer Science Logic, October, 1987, Karlsruhe FRG, Lecture Notes in Computer Science, vol. 329, Springer-Verlag, 1987.
[14] D. Harel, Dynamic Logic (in [7]).
[15] D. Harel, First Order Dynamic logic, Lecture Notes in Computer Science, vol. 68, 1976.
[16] C. Hoare, An axiomatic basis for computer programming, Communications of the ACM 12 (1969) 576–583.
[17] D. Kozen, R. Parikh, An elementary proof of the completeness of PDL, Theoretical Computer Science 14 (1981) 113–118.
[18] D. Kozen, J. Tiuryn, Logics of Programs, Technical Report 89-962, Dept. of Computer Science, Cornell University 1989.
[19] D. Leivant, Proof Theoretic Methodology for Propositional Dynamic Logic, Lecture Notes in Computer Science, vol. 107, 1981.
[20] A. Nerode, Some lectures in modal logic, Technical Report, M.S.I. Cornell University, 1989 CIME Logic and Computer Science Montecatini Volume, Springer-Verlag Lecture Notes, 1990.
[21] A. Nerode, Some lectures in intuitionistic logic, Technical Report, M.S.I. Cornell University, 1988, Marktoberdorf Logic and Computation NATO Summer School Volume, NATO Science Series, 1990 (in press).
[22] H. Nishimura, Semantical analysis of constructive PDL, Publications of the Research Institute of Mathematical Sciences 18 (2) (1982).
[23] D. Peleg, Concurrent dynamic logic, Journal of the Association for Computing Machinery 34 (2) (1987).
[24] D. Peleg, Communication in concurrent dynamic logic, Journal of Computer and System Sciences 35 (1987).
[25] G. Plotkin, C. Stirling, A framework for intuitionistic modal logic, in: J.Y. Halpern (Ed.), Theoretical Aspects of Reasoning About Knowledge, 1986.
[26] V. Pratt, Semantical considerations on Floyd–Hoare logic, in: 17th Annual IEEE Symp. on Found. Comp. Sci., New York, 1976, pp. 109–121.
[27] V. Pratt, Applications of modal logic to programming, Studia Logica 39 (1980) 257–274.
[28] W. Reif, Vollständigkeit einer modifizierten Goldblatt-Logik und Approximation der Omegaregel durch Induktion, Diplomarbeit, Facultät für Informatik, Universität Karlsruhe, 1986.
[29] W. Reif, A completeness result in non-standard dynamic logic, Interner Bericht 5/85, Facultät für Informatik, Universität Karlsruhe, 1986.
[30] A. Troelstra, D. Van Dalen, Constructivism in Mathematics: An Introduction I, II, North Holland, 1988.
[31] D. Wijesekera, Constructive modal logic I, Annals of Pure and Applied Logic 50 (1990) 271–301.