Logical Methods in Computer Science
Vol. 12(3:6)2016, pp. 1–41
www.lmcs-online.org

Submitted Nov. 30, 2015
Published Sep. 5, 2016

SYNTHESIS OF STRATEGIES USING THE HOARE LOGIC OF ANGELIC AND DEMONIC NONDETERMINISM *

KONSTANTINOS MAMOURAS

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA
e-mail address: mamouras@seas.upenn.edu

Abstract. We study a propositional variant of Hoare logic that can be used for reasoning about programs that exhibit both angelic and demonic nondeterminism. We work in an uninterpreted setting, where the meaning of the atomic actions is specified axiomatically using hypotheses of a certain form. Our logical formalism is entirely compositional and it subsumes the non-compositional formalism of safety games on finite graphs. We present sound and complete Hoare-style calculi that are useful for establishing partial-correctness assertions, as well as for synthesizing implementations. The computational complexity of the Hoare theory of dual nondeterminism is investigated using operational models, and it is shown that the theory is complete for exponential time.

1. Introduction

Demonic nondeterminism is used in the context of programming to model external influences which are not under the control of the program. Such nondeterminism may arise in concurrent programs, for example, from the scheduling of threads, which is under the control of the operating system and not the program. Others examples could be sensor readings or user input, which are completely external influences to a computing system. In the case of user input, in particular, we can typically make no assumptions, since the input depends on an entirely unpredictable and uncontrollable human being, who may choose to behave as an adversary.

Even in the absence of "real" nondeterminacy like scheduling and sensor/user input, we may use demonic nondeterminism to represent abstraction and partial knowledge of the state of a computation. An example of the latter use of demonic nondeterminism is when we cannot fully observe the value of an integer variable x, but we can tell whether it is negative, zero, or positive. At this level of abstraction, we cannot describe the operation

2012 ACM CCS: [**Theory of computation**]: Logic—Logic and verification; [**Software and its engineering**]: Software organization and properties—Software functional properties—Formal methods—Software verification.

Key words and phrases: Hoare logic, program synthesis, angelic and demonic nondeterminism, safety games, program schemes, dual nondeterminism.

* This is a revised and expanded version of the paper [Mam15b], which was presented in FoSSaCS 2015.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-12(3:6)2016

$x := x + 1$ that increments the variable x by 1 deterministically.

Observe x fully		**Observe x partially**
deterministic action	$\xrightarrow{\text{abstraction}}$	corresponding nondeterministic action
$x \mapsto x + 1$		$(x < 0) \mapsto (x < 0) \vee (x = 0)$
		$(x = 0) \mapsto (x > 0)$ and $(x > 0) \mapsto (x > 0)$

This example illustrates that nondeterminism is necessary when creating finite-state abstractions of realistic programs, whose state space is typically infinite.

Angelic nondeterminism, on the other hand, is used to express nondeterminacy that is under the control of the program. We use angelic nondeterminism to leave some implementation details of a program underspecified. The "angel", namely the agent that represents our interests, controls how these details are resolved in order to achieve the desired result. The process of resolving these implementation details amounts to *synthesizing* a fully specified program. The term *dual nondeterminism* refers to the combination of angelic and demonic nondeterminism.

In order to reason about dual nondeterminism, one first needs to have a semantic model of how programs with angelic and demonic choices compute. One semantic model that has been used extensively uses a class of mathematical objects that are called monotonic predicate transformers [BW98] (based on Dijkstra's predicate transformer semantics [Dij75, Mor98]). An equivalent denotational model that is based on binary relations was introduced in [Rew03] (up-closed multirelations) and further investigated in [MCR04, MCR07, MC13]. These relations can be understood intuitively as two-round games between the angel and the demon.

We are interested here in verifying properties of programs that can be expressed as Hoare (partial-correctness) assertions [Flo67, Hoa69, Coo78, Apt81, Apt83], that is, formulas of the form $\{p\}f\{q\}$, where f is the program text and p, q denote predicates on the state space, called precondition and postcondition respectively. The formula $\{p\}f\{q\}$ asserts, informally, that starting from any state satisfying the precondition p, the angel has a strategy so that whatever the demon does, the final state of the computation of f (assuming termination) satisfies the postcondition q. This describes a notion of partial correctness, because in the case of divergence (non-termination) the angel wins vacuously. Our language for programs and preconditions/postconditions involves abstract test symbols $p, q, r, \ldots$ and abstract action symbols $a, b, \ldots$ with no fixed interpretation. We constrain their meaning with extra hypotheses: we consider a finite set Φ of Boolean axioms for the tests, and a finite set Ψ of axioms of the form $\{p\}a\{q\}$ for the action letters. So, we typically assert implications of the form

$$\Phi, \Psi \Rightarrow \{p\}f\{q\},$$

which we call *simple Hoare implications*. For example, consider the tests $even(n)$, $odd(n)$ and the action $n{+}{+}$, which increments n by 1. We think that these are abstract symbols contrained by the hypotheses Φ and Ψ below.

$$\Phi : \begin{array}{l} even(n) \vee odd(n) \\ \neg even(n) \vee \neg odd(n) \end{array} \qquad \Psi : \begin{array}{c} \{even(n)\}n{+}{+}\{odd(n)\} \\ \{odd(n)\}n{+}{+}\{even(n)\} \end{array} \qquad f := \begin{array}{l} \mathsf{if}\ even(n)\ \mathsf{then}\ n{+}{+} \\ \mathsf{else}\ n{+}{+}; n{+}{+} \end{array}$$

We should be able to prove that $\Phi, \Psi \Rightarrow \{\mathsf{true}\}f\{odd(n)\}$ under the above definitions. We want to design a formal system that allows the derivation of the valid Hoare implications. One important desideratum for such a formal system is to also provide us with program

text that corresponds to the winning strategy of the angel. Then, the system can be used for the deductive synthesis of programs that satisfy their Hoare specifications.

There has been previous work on deductive methods to reduce angelic nondeterminism and synthesize winning strategies for the angel. The work [CvW03], which is based on ideas of the refinement calculus [BvW90, BvW92, BW98, Mor98], explores a total-correctness Hoare-style calculus to reason about angelic nondeterminism. It is observed that there is a conceptual difficulty in reconciling *nondeterministic refinement* (which results from removing demonic choices or/and adding angelic choices) with the task of synthesizing the strategy of the angel. This is because the interaction between the angel and the demon has been fixed in advance: we have no control over the demonic nondeterminism, and increasing the choices of the angel is not permitted. Nonetheless, a refinement-based approach for implementing angelic choices is pursued in [CvW03]. The analysis is in the first-order interpreted setting, and no completeness or relative completeness results are discussed.

Of particular relevance to our investigations is the line of work that concerns two-player infinite games played on finite graphs [Tho95]. Such games are useful for analyzing (nonterminating) reactive programs. One of the players represents the "environment", and the other player is the "controller". Computing the strategies that witness the winning regions of the two players amounts to synthesizing an appropriate implementation for the controller. The formalism of games on finite graphs is very convenient for developing an algorithmic theory of synthesis. However, the formalism is non-succinct and, additionally, it is inherently non-compositional. An important class of properties for these graph games are the so called *safety* properties, which assert that the environment cannot force the play into a "bad" region. For encoding safety properties, we see that a fully compositional formalism based on while programs and partial-correctness properties suffices.

Our Contribution. We consider a propositionally abstracted language for while programs with demonic and angelic choices. Our results are the following:

- We give the intended operational semantics in terms of safety games on graphs, and we describe a denotational semantics based on a restricted subclass of multirelations. We obtain a full abstraction result for all reasonable intepretations of the atomic symbols, which asserts the equivalence between the operational and denotational models.
- We present a sound and *unconditionally* complete calculus for the weak Hoare theory of dual nondeterminism (over the class of all interpretations). We also consider a restricted class of interpretations, where the atomic actions are non-angelic, and we extend our calculus so that it is complete for the Hoare theory of this smaller class (called strong Hoare theory). The proofs of these results rely on the construction of free models.
- Using the correspondence between the operational and denotational models, we prove that the strong Hoare theory of dual nondeterminism is EXPTIME-complete.
- We consider an extension of our Hoare-style calculus with annotations that denote the winning strategies of the angel. We thus obtain a sound and complete deductive system for the synthesis of angelic strategies.
- Our formalism is shown to subsume that of safety games on finite graphs, hence it provides a compositional method for reasoning about safety in reactive systems. The language of dually nondeterministic program schemes is exponentially more succinct than explicitly represented game graphs, and it is arguably a more natural language for describing algorithms and protocols.

The present paper is a revised and extended version of [Mam15b]. We include here all the proofs that were omitted from the conference version [Mam15b], and we generalize the full abstraction result on the correspondence between the operational and denotational semantics. In [Mam15b], full abstraction was established only for the free models, which are finite. In order to generalize the full abstraction theorem to infinite models, we identify here a natural condition on the interpretations of atomic actions (which we call *chain property*). This condition covers all finite models, as well as all infinite models with a "reasonable" interpretation of the atomic actions.

Outline of paper. In §2 we recall some well-known definitions and facts about abstract imperative while programs, and we introduce the relevant notation that we will use in our later development. We introduce *while game schemes* in §3, which are abstractions of programs that allow both angelic and demonic nondeterministic choices. We also present in §3 the *intended operational semantics*, which is based on the familiar model of two-player safety games played on graphs. We explore in §4 a denotational model based on a certain kind of binary relations. We show that this denotational semantics extends naturally the standard relational semantics of programs, and additionally it agrees exactly with the intended operational model. In §5 we introduce the syntax and meaning of Hoare assertions and implications, and we propose a Hoare-style calculus for reasoning about while game schemes. Our first completeness result is given in §6, where we show that the partial-correctness calculus of §5 is complete for the *weak Hoare theory* (the theory over the class of all interpretations). In §7 we study the *strong Hoare theory*, which is the theory over the subclass of interpretations that assign a non-angelic meaning to the atomic actions. We extend our calculus to completeness for this important case, and we show that the theory is complete for EXPTIME. We further extend in §8 our axiomatization of the strong Hoare theory with annotations that witness the angelic strategies. We thus obtain a sound and complete Hoare-style calculus for the synthesis of angelic implementations. It is also shown that our formalism subsumes the (non-compositional and non-succinct) formalism of safety games on finite graphs. We analyze a simple example in §9 for a toy temperature controller, which illustrates in a very concrete way how our verification/synthesis calculus can be used. In §10 we discuss several related works, including the ones from which the present paper was inspired. We conclude in §11 with a brief summary of our technical contribution, and with suggestions for future work.

2. Preliminaries: Monadic While Program Schemes

In this section we give some preliminary definitions regarding abstract imperative programs with while loops, which are also known in the literature as *while program schemes*. See for example [Rut64, Pat68, LPP70, PH70, GL73] for some very well-known works in the area of program schematology. The programs that we consider here are often qualified as *monadic*, which means that the program state is considered to be one indivisible entity. In other words, the program actions are modeled as unary functions that act on the entire program state. There are no distinct program variables $x, y, z, \ldots$ at the syntactic level, nor variable assignments $z \leftarrow f(x, y)$ that can read from and assign to variables individually. Instead, the primitive actions are written simply as atomic letters $a, b, c, \ldots$ that should be thought as transforming the whole program state. Alternatively, one can think equivalently

that there is a single program variable x (which represents the entire program state) and an atomic action a corresponds to an assignment $x \leftarrow a(x)$.

We are interested in program schemes that allow the use of the construct $\sqcap$ of demonic nondeterministic choice. This is a very useful operation, because it can model underspecification and real nondeterminism (environment, user input, and so on). First, we present the syntax of these abstract while programs. Then, we give the standard denotational semantics for them, which is based on binary relations.

Definition 2.1 (The Syntax of Program Schemes)**.** We consider a two-sorted algebraic language. There is the sort of *tests* and the sort of *programs*. The tests are built up from *atomic tests* and the constants true and false, using the usual Boolean operations: $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction). We use the letters $p, q, r, \ldots$ to range over arbitrary tests. Tests are thus given by the grammar:

$$\text{tests } p, q ::= \text{atomic test} \mid \mathsf{true} \mid \mathsf{false} \mid \neg p \mid p \wedge q \mid p \vee q.$$

As usual, the implication $p \to q$ is abbreviation for $\neg p \vee q$, and the double implication $p \leftrightarrow q$ stands for $(p \to q) \wedge (q \to p)$.

The base programs are the *atomic programs* $a, b, c, \ldots$ (also called *atomic actions*), as well as the constants id (*skip*) and $\bot$ (*diverge*). The programs are constructed using the operations ; (*sequential composition*), if (*conditional*), while (*iteration*), and $\sqcap$ (*demonic nondeterministic choice*). We write $f, g, h, \ldots$ to range over arbitrary programs. So, the programs are given by the following grammar:

$$\begin{aligned}\text{programs } f, g ::= \text{atomic actions } a, b, c, \ldots \mid \mathsf{id} \mid \bot \mid \\ f; g \mid \mathsf{if}\, p\, \mathsf{then}\, f\, \mathsf{else}\, g \mid \mathsf{while}\, p\, \mathsf{do}\, f \mid f \sqcap g.\end{aligned}$$

For brevity, we also write $p[f, g]$ instead of $\mathsf{if}\, p\, \mathsf{then}\, f\, \mathsf{else}\, g$, and $\mathsf{W}pf$ instead of $\mathsf{while}\, p\, \mathsf{do}\, f$.

In order to give meaning to these abstract while programs, we first need to specify a nonempty set S representing the state space. Additionally, we need to know how the atomic actions $a, b, c, \ldots$ transform the program state, and which states satisfy an atomic test p. So, for every atomic test we are given a subset $R(p) \subseteq S$ of the states that satisfy p. Moreover, for every action a assume that we are given a function $R(a) : S \to \wp S$, where $\wp S$ is the powerset of S. If u and v are states in S with $v \in R(a)(u)$, then we understand this as saying that: executing the action a when in state u may result in a final state v. It remains now to describe how an arbitrary program scheme computes. The intended semantics is *operational* and it gives us all the intermediate steps of the computation. A configuration is a pair (u, f) of a state u and a program f and $\to$ is a relation on configurations that describes one step of the computation. A configuration (u, id) is *final*, which means that the computation halts. We see in Figure 1 the standard definition of the computation relation, where we have assumed w.l.o.g. that ; is associative.

The operational semantics of Figure 1 describes fully how a program executes, but for our later logical investigation this description carries too much irrelevant information. We would instead like to focus on the *input-output* behavior of a program f. We thus *summarize* the meaning of f as a function $R(f) : S \to \wp S$, which is defined as follows:

$$v \in R(f)(u) \overset{\text{def}}{\iff} (u, f) \to \cdots \to (v, \mathsf{id}).$$

$$
\begin{array}{rlrl}
(u,a) &\to (v,\mathsf{id}),\ \text{for } v\in R(a)(u) & (u,a;h) &\to (v,\mathsf{id};h),\ \text{for } v\in R(a)(u)\\
(u,\mathsf{id}) &\to & (u,\mathsf{id};h) &\to (u,h)\\
(u,\bot) &\to (u,\bot) & (u,\bot;h) &\to (u,\bot;h)\\
(u,p[f,g]) &\to (u,f),\ \text{if } u\in R(p) & (u,p[f,g];h) &\to (a,f;h),\ \text{if } u\in R(p)\\
(u,p[f,g]) &\to (u,g),\ \text{if } u\notin R(p) & (u,p[f,g];h) &\to (a,g;h),\ \text{if } u\notin R(p)\\
(u,\mathsf{W}pf) &\to (u,f;\mathsf{W}pf),\ \text{if } u\in R(p) & (u,(\mathsf{W}pf);h) &\to (u,f;(\mathsf{W}pf);h),\ \text{if } u\notin R(p)\\
(u,\mathsf{W}pf) &\to (u,\mathsf{id}),\ \text{if } u\notin R(p) & (u,(\mathsf{W}pf);h) &\to (u,\mathsf{id};h),\ \text{if } u\notin R(p)\\
(u,f\sqcap g) &\to (u,f),\ (u,g) & (u,(f\sqcap g);h) &\to (u,f;h),\ (\alpha,g;h)
\end{array}
$$

Figure 1: While Program Schemes: The standard operational model for the interpretation R of atomic symbols.

The right-hand side of the above equivalence says that there is a sequence of computation steps from the initial configuration (u,f) to the final configuration (id,v). These input-output summaries $R(f):S\to\wp S$ constitute the standard *denotational semantics* of nondeterministic while program schemes, also known as the *relational semantics* of programs. It is a very pleasant fact that the functions $R(f)$ have a straightforward *compositional* definition, namely by induction on the structure of f. This result is completely standard, and it asserts that denotational equality coincides with operational equivalence. This property is sometimes dubbed as *full abstraction*.

Before we give the formal denotational semantics of while program schemes, we need to define some useful notation. In particular, we will consider an algebra of binary relations (equivalently, their representation as "nondeterministic functions") with operations that can give direct meaning to the syntactic constructors of program schemes.

Definition 2.2 (Nondeterministic Functions & Operations)**.** For a set S, we write $\wp S$ to denote the *powerset* of S. A function of type $k:S\to\wp S$ is a *nondeterministic function* on S. We also use the notation $k:S\rightsquigarrow S$. We write $k:u\mapsto v$ to mean that $v\in k(u)$. We think informally that such a function describes only one kind of nondeterminism (for our purposes here, demonic nondeterminism). Consider the operations of Figure 2. The choice operation $+$ induces a partial order $\leq$ on $S\rightsquigarrow S$ given by : $k\leq\ell$ iff $k+\ell=\ell$.

Definition 2.3 (Nondeterministic Interpretation of Program Schemes)**.** An interpretation of the language of nondeterministic while program schemes consists of a nonempty set S, called the *state space*, and an *interpretation function* R. The elements of S are called *states*, and we will be using letters $u,v,w,\ldots$ to range over them. For a program term f, its *interpretation* $R(f):S\rightsquigarrow S$ is a nondeterministic function on S.

The interpretation $R(p)$ of a test p is a unary predicate on S, i.e., $R(p)\subseteq S$. R specifies the meaning of every atomic test, and it extends as follows:

$$
\begin{array}{lll}
R(\mathsf{true})=S & R(\neg p)=\sim R(p) & R(p\wedge q)=R(p)\cap R(q)\\
R(\mathsf{false})=\emptyset & & R(p\vee q)=R(p)\cup R(q)
\end{array}
$$

where $\sim$ is the operation of complementation w.r.t. S, that is, $\sim A=S\setminus A$. Moreover, the interpretation function R specifies the meaning $R(a):S\rightsquigarrow S$ of every atomic program. We

(Kleisli) composition ;	$(k;\ell)(u) \triangleq \bigcup_{v\in k(u)} \ell(v)$
Conditional $(\cdot)[-,-]$	$P[k,\ell](u) \triangleq k(u),\ \text{if } u \in P$
	$P[k,\ell](u) \triangleq \ell(u),\ \text{if } u \notin P$
Binary choice $+$	$(k+\ell)(u) \triangleq k(u) \cup \ell(u)$
Arbitrary choice $\sum$	$\big(\sum_{i\in J} k_i\big)(u) \triangleq \bigcup_{i\in J} k_i(u)$
Identity 1_S	$1_S(u) \triangleq \{u\}$
Zero 0_S	$0_S(u) \triangleq \emptyset$
Iteration $(\text{wh}\ \cdot\ \text{do}\ -)$	$\text{wh}\, P \,\text{do}\, k \triangleq \sum_{n\geq 0} V_n,\ \text{where}$
	$V_0 \triangleq P[0_S, 1_S]$
	$V_{n+1} \triangleq P[k; V_n, 1_S]$

Figure 2: Semantic operations for nondeterministic functions $S \rightsquigarrow S$.

extend the interpretation to all program terms:

$$R(\mathsf{id}) = 1_S \qquad R(f;g) = R(f);R(g) \qquad R(p[f,g]) = R(p)[R(f),R(g)]$$
$$R(\bot) = 0_S \qquad R(f \sqcap g) = R(f) + R(g) \qquad R(\mathsf{w}pf) = \text{wh}\, R(p)\,\text{do}\, R(f)$$

Our definition agrees with the standard relational semantics of while schemes.

3. The Operational Semantics of Dual Nondeterminism

We extend the syntax of nondeterministic program schemes with the additional construct $\sqcup$ of *angelic (nondeterministic) choice*. So, the grammar for the program terms now becomes:

$$\text{programs } f,\, g ::= \text{actions } a, b, \ldots \mid \mathsf{id} \mid \bot \mid f;g \mid p[f,g] \mid \mathsf{w}pf \mid f \sqcap g \mid f \sqcup g.$$

We call these program terms *while game schemes*, because they can be considered to be descriptions of games between the angel (who controls the angelic choices) and the demon (who controls the demonic choices). Informally, the angel tries to satisfy the specification, while the demon attempts to falsify it.

We consider two-player games between the *existential* player $\exists$ (angel) and the *universal* player $\forall$ (demon). The games are played on arenas of arbitrary cardinality and are of infinite duration. If σ is a player, then $\neg\sigma$ is the other player. Such games are considered extensively in the literature for the verification of reactive systems, see for example [Tho95]. The following definition of safety games (Definition 3.1) slightly modifies the definition of [Tho95] in order to fit our setting more naturally.

Definition 3.1 (Safety Games)**.** A *safety game* is a tuple $G = (V, V_\exists, V_\forall, \to, E)$, where V is the set of all vertices, $V_\exists$ is the set of $\exists$-vertices (which belong to the existential player), $V_\forall$ is the set of $\forall$-vertices (which belong to the universal player), $V_\exists$ and $V_\forall$ are disjoint subsets of V, $\to$ is a binary *transition relation* on V, and $E \subseteq V$ is the set of *error vertices*. We use the letters $u, v, w, \ldots$ to range over vertices in V, and we write $u \to v$ to mean that the pair (u, v) belongs to the transition relation. We require additionally that every vertex has a successor, and that the vertices $V_? = V \setminus (V_\exists \cup V_\forall)$ that belong to no player have exactly

one successor. The last requirement says equivalently that if a vertex has more than one successor, then it must belong to one of the players.

We need to introduce some terminology, which is to be understood with respect to a specific game. A *position* is a finite nonempty path, and a *play* is an infinite path. A u-position (u-play) is a position (play) that starts from vertex u. We say that Player $\exists$ *wins* a play if no error vertex appears in it. Player $\forall$ wins if the play contains an error vertex. A *strategy for Player* σ or a σ*-strategy* is a function that maps every position ending in a σ-vertex u to one of the successors of u. In a *memoryless* or *positional* strategy the choice depends only on the last vertex. So, we can represent a memoryless strategy for Player σ as a function that maps every σ-vertex to one of its successors. We say that a path *conforms* to a σ-strategy f_σ if every transition from a σ-vertex in the path is the one prescribed by the strategy f_σ. A (u, f_σ)-position is a u-position that conforms to the strategy f_σ. We define a (u, f_σ)-play similarly. A $(u, f_\exists, f_\forall)$-position is a u-position that conforms to both $f_\exists$ and $f_\forall$. A $(u, f_\exists, f_\forall)$-play is defined similarly. We denote by $\mathsf{play}(u, f_\exists, f_\forall)$ the unique $(u, f_\exists, f_\forall)$-play, which is the infinite path formed by starting at vertex u and then following the strategies $f_\exists$ and $f_\forall$ for every transition allowing more than one choice.

We say that a set of vertices $U \subseteq V$ is σ*-closed* if

(i) every vertex of $V_? \cap U$ has its unique successor in U,
(ii) every σ-vertex of U has at least one successor in U, and
(iii) every $\neg\sigma$-vertex of U has all of its successors in U.

Definition 3.2 (Winning Regions)**.** Given a safety game $G = (V, V_\exists, V_\forall, \to, E)$, we will define the sets $W_\exists \subseteq V$ and $W_\forall \subseteq V$, which partition the set V of vertices. The set $W_\exists$ is called the *winning region* of Player $\exists$, and $W_\forall$ is the *winning region* of Player $\forall$. First, we define the transfinite sequence $(W_\forall^\kappa)_{\kappa \in \mathbf{Ord}}$ of sets. We write **Ord** for the class of ordinals. Informally, for an ordinal κ, the set $W_\forall^\kappa$ consists of the nodes from which Player $\forall$ can force a visit to E in at most κ steps.

$$W_\forall^0 \triangleq E \qquad W_\forall^{\kappa+1} \triangleq W_\forall^\kappa \cup \{u \in V_? \mid \text{the unique successor of } u \text{ is in } W_\forall^\kappa\} \cup \{u \in V_\exists \mid \text{every successor of } u \text{ is in } W_\forall^\kappa\} \cup \{u \in V_\forall \mid \text{some successor of } u \text{ is in } W_\forall^\kappa\}$$

$$W_\forall^\lambda \triangleq \bigcup_{\kappa<\lambda} W_\forall^\kappa, \text{ for a limit ordinal } \lambda$$

Now, we can define the winning regions of the players in terms of the above sequence:

$$W_\forall \triangleq \bigcup_{\kappa \in \mathbf{Ord}} W_\forall^\kappa \qquad\qquad W_\exists \triangleq V \setminus W_\forall$$

Notice that the sets $W_\forall^0 \subseteq W_\forall^1 \subseteq \cdots \subseteq W_\forall^\kappa \subseteq \cdots$ form a transfinite chain w.r.t. inclusion.

Theorem 3.3 (Memoryless Determinacy)**.** *Let $G = (V, V_\exists, V_\forall, \to, E)$ be a safety game, and $W_\exists$, $W_\forall$ be the winning regions of the two players. There is a memoryless $\exists$-strategy $f_\exists^*$ and a memoryless $\forall$-strategy $f_\forall^*$ that witness uniformly the winning regions. That is:*

(1) *For every $u \in W_\exists$ and every $\forall$-strategy $f_\forall$, $\mathsf{play}(u, f_\exists^*, f_\forall)$ is won by Player $\exists$.*
(2) *For every $u \in W_\forall$ and every $\exists$-strategy $f_\exists$, $\mathsf{play}(u, f_\exists, f_\forall^*)$ is won by Player $\forall$.*

Proof sketch. The idea for Part (1) is to show that the set $W_\exists$ is $\exists$-closed, and therefore Player $\exists$ has a memoryless strategy $f_\exists^*$ that keeps within $W_\exists$ every play starting from a vertex of $W_\exists$. For the sake of contradiction, assume that $u \in W_\exists$ is a vertex which witnesses that $W_\exists$ is *not* $\exists$-closed. There are three distinct possibilities for u:

(i) $u \in V_?$ and its unique successor is in $W_\forall$, or
(ii) $u \in V_\exists$ and every successor of u is in $W_\forall$, or
(iii) $u \in V_\forall$ and some successor of u is in $W_\forall$.

Every possibility implies that $u \in W_\forall$, which gives the desired contradiction. So, $W_\exists$ is indeed $\exists$-closed. For Part (2), the proof is based on labeling every vertex $u \in W_\forall$ as follows:

$$\mathsf{ord}(u) \triangleq \text{the least ordinal } \kappa \text{ such that } u \in W_\forall^\kappa.$$

One can then show that Player $\forall$ has a strategy $f_\forall^*$ so that for every play that starts from a vertex of $W_\forall$ the labels keep going down until eventually an error vertex is reached. □

Observation 3.4 (Summarizing Safety Games)**.** We have already discussed in §2 that a denotational semantics is most useful when it is a *faithful summarization* of the intended operational meaning. Before presenting a denotational semantics of dual nondeterminism in §4 we will discuss here what constitutes a summarization for safety games, and what kind of mathematical objects are useful for this purpose.

Consider a safety game $(V, V_\exists, V_\forall, \to, E)$ and recall that $W_\exists$ is the set of vertices from which the existential player (angel) has a strategy to avoid the error vertices. We write $W_\exists(E)$ to emphasize the fact that the winning region of Player $\exists$ depends on which vertices are designated as error vertices. Theorem 3.3 implies that:

If $u \in W_\exists(E)$ then the angel can keep any u-play within the non-error vertices $\sim E$.

Let us think about the more general situation, where the error vertices E can be varied. We can summarize the guarantees that the angel can make with the following object:

$$\phi \triangleq \{(u, \sim E) \mid \text{in the game } (V, V_\exists, V_\forall, \to, E), \text{ the vertex } u \text{ is in } W_\exists(E)\}.$$

Immediately from the definition of the winning regions (see Definition 3.2) we see that:

(1) The inclusion $E_1 \subseteq E_2$ implies $W_\forall(E_1) \subseteq W_\forall(E_2)$ and therefore $W_\exists(E_2) \subseteq W_\exists(E_1)$. Assuming that $X \subseteq Y \subseteq V$ we have that $\sim Y \subseteq \sim X$ and

$$(u, X) \in \phi \implies u \in W_\exists(\sim X) \implies u \in W_\exists(\sim Y) \implies (u, Y) \in \phi.$$

(2) Notice that for error vertices $E = \emptyset$ we have that $W_\forall(\emptyset) = \emptyset$ and hence $W_\exists(\emptyset) = V$. It follows that (u, V) belongs to ϕ.

Both of the above properties will turn out to be crucial for our development, and they motivate the notion of a *game function* given formally in Definition 4.1 of §4. For the rest of this section, it suffices to keep in mind that the denotations of game schemes will be binary relations from S to $\wp S$, where S is the state space.

In order to streamline the presentation of the operational semantics, we should make a couple of inconsequential modifications to the language of game schemes. We restrict slightly the syntax of program terms by eliminating the diverging $\bot$ program, and by forbidding compositions $(f; g); h$ that associate to the left. These are not really limitations, because for every reasonable semantics $\bot$ has to be equivalent to the infinite loop $\mathsf{while}\,\mathsf{true}\,\mathsf{do}\,\mathsf{id}$, and $(f; g); h$ has to be equivalent to $f; (g; h)$. So, we define the syntactic categories *factor* and *term* with the following grammars:

$$\text{factor } e ::= \text{atomic program } a, b, \ldots \mid \mathsf{id} \mid p[f, g] \mid \mathsf{W}pf \mid f \sqcup g \mid f \sqcap g$$
$$\text{terms } f, g ::= e \mid e; f$$

According to the above definition, a term is a nonempty list of factors. We write @ for the operation of list concatenation: $e@g = e; g$ and $(e; f)@g = e; (f@g)$.

Definition 3.5 (Closure & The $\rightarrowtriangle$ Relation On Terms)**.** We define the *closure* map $C(\cdot)$, which sends a program term to a finite set of program terms.

$$C(a) = \{a, \mathsf{id}\} \qquad C(\mathsf{w}pf) = \{\mathsf{w}pf, \mathsf{id}\} \cup C(f)@\mathsf{w}pf \qquad C(e; f) = C(e)@f \cup C(f)$$
$$C(\mathsf{id}) = \{\mathsf{id}\} \qquad C(f \oplus g) = \{f \oplus g\} \cup C(f) \cup C(g)$$

where $\oplus$ is any of the constructors $\sqcup$, $\sqcap$, or $p[-,-]$. If F is a set of terms and g is a term, we lift the concatenation operation $@$ as follows: $F@g = \{f@g \mid f \in F\}$. Now, we define the *one-step reachability relation* $\rightarrowtriangle$ on program terms as follows:

$$a \rightarrowtriangle \mathsf{id} \qquad a; h \rightarrowtriangle \mathsf{id}; h$$
$$\mathsf{id} \rightarrowtriangle \qquad \mathsf{id}; h \rightarrowtriangle h$$
$$f \oplus g \rightarrowtriangle f,\, g \qquad (f \oplus g); h \rightarrowtriangle f@h,\, g@h$$
$$\mathsf{w}pf \rightarrowtriangle f@\mathsf{w}pf,\, \mathsf{id} \qquad \mathsf{w}pf; h \rightarrowtriangle f@(\mathsf{w}pf); h,\, \mathsf{id}; h$$

The above definition of $\rightarrowtriangle$ says, in particular, that id has no successor. The while loop $\mathsf{w}pf$ has exactly two successors, namely $f@\mathsf{w}pf$ and id. We write $\rightarrowtriangle^*$ to denote the reflexive transitive closure of the relation $\rightarrowtriangle$.

Lemma 3.6. The following hold for the closure map and the reachability relation:

(1) Let f be a program term. The cardinality of the set $C(f)$ is linear in the size $|f|$ of the term f. More specifically, it holds that $|C(f)| \leq 2|f|$.
(2) For terms f, f' and g, if $f \rightarrowtriangle f'$ then $f@g \rightarrowtriangle f'@g$.
(3) For terms f, f' and g, if $f \rightarrowtriangle^* f'$ then $f@g \rightarrowtriangle^* f'@g$.
(4) For terms f and g, the $\rightarrowtriangle$-successors of $f@g$ are contained in $\{g\} \cup \{f'@g \mid f \rightarrowtriangle f'\}$.
(5) For every term f, the set $C(f)$ contains f and is closed under $\rightarrowtriangle$.
(6) For all terms f and f', if $f' \in C(f)$ then $f \rightarrowtriangle^* f'$.
(7) Let f be a program term. Then, $C(f)$ is equal to the set $\{f' \mid f \rightarrowtriangle^* f'\}$ of terms that are reachable from f via $\rightarrowtriangle$.

Note: Parts (1) and (7) are the main properties that we will need later. Parts (2)–(6) are the intermediate claims that are needed to obtain Part (7).

Proof. Part (1) can be shown by induction on the structure of f. Parts (2) and (4) are proved with a case analysis on the form of the term f. Part (3) follows from Part (2) by induction on the length of the $\rightarrowtriangle$-sequence. Part (5) is shown by induction on f, making use of Part (4). The proof of Part (6) requires an induction on f and Part (3). Part (7) is an immediate consequence of Part (5) and Part (6). □

Definition 3.7 (Operational Model for Game Schemes)**.** Let S be a nonempty set of states, and I be an interpretation function for the atomic tests and actions. That is, I specifies a unary predicate $I(p) \subseteq S$ for every atomic test p, and a binary relation $I(a) \subseteq S \times \wp S$ for every atomic action a. Let f be a program term, and $E \subseteq S$ be a set of *error states*. We define the *operational model* for I, f, E, denoted $G_I(f, E)$, to be the safety game

$$G_I(f, E) = (V, V_\exists, V_\forall, \rightarrow, E \times \{\mathsf{id}\}), \text{ where}$$
$$V = (S \times C(f)) \cup (\mathcal{X} \times C(f)) \text{ with}$$
$$\mathcal{X} = \{X \subseteq S \mid (u, X) \in I(a) \text{ for some } a \in C(f) \text{ and } u \in S\},$$

and the transition relation $\rightarrow$ is defined in Figure 3. Part (7) of Lemma 3.6 implies that V is closed under $\rightarrow$ (note that $\rightarrowtriangle$ is the "projection" of $\rightarrow$ to the second component). Strictly

$$
\begin{array}{rl}
(u,a) \to (X,\mathsf{id}), \text{ when } (u,X)\in I(a) & (u,a;h) \to (X,\mathsf{id};h), \text{ when } (u,X)\in I(a)\\
(u,\mathsf{id}) \to & (u,\mathsf{id};h) \to (u,h)\\
(u,p[f,g]) \to (u,f), \text{ if } u\in I(p) & (u,p[f,g];h) \to (a,f@h), \text{ if } u\in I(p)\\
(u,p[f,g]) \to (u,g), \text{ if } u\notin I(p) & (u,p[f,g];h) \to (a,g@h), \text{ if } u\notin I(p)\\
(u,\mathsf{W}pf) \to (u,f@\mathsf{W}pf), \text{ if } u\in I(p) & (u,(\mathsf{W}pf);h) \to (u,f@(\mathsf{W}pf);h), \text{ if } u\notin I(p)\\
(u,\mathsf{W}pf) \to (u,\mathsf{id}), \text{ if } u\notin I(p) & (u,(\mathsf{W}pf);h) \to (u,\mathsf{id};h), \text{ if } u\notin I(p)\\
(u,f\sqcup g) \to (u,f),\ (u,g) & (u,(f\sqcup g);h) \to (u,f@h),\ (u,g@h)\\
(u,f\sqcap g) \to (u,f),\ (u,g) & (u,(f\sqcap g);h) \to (u,f@h),\ (u,g@h)
\end{array}
$$

$$(X,f) \to (v,f), \text{ where } v\in X\subseteq S$$

Figure 3: While Game Schemes: Operational model for interpretation I of atomic symbols.

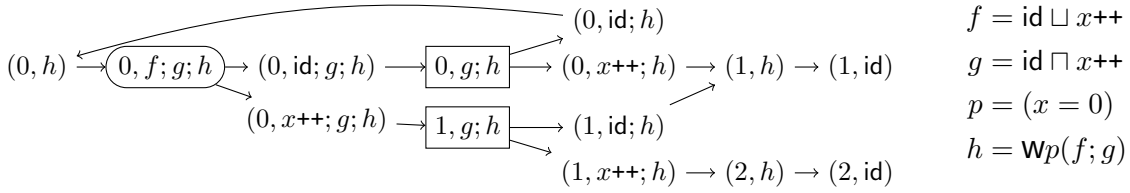


Figure 4: Reduced operational model for the dually nondeterministic program h. The vertices of the demon (angel) are indicated with rectangles (rounded rectangles).

speaking, in order for $G_I(f,E)$ to be a safety game according to Definition 3.1, we would need to modify $\to$ so that every vertex (u,id) has a self-loop instead of being a sink, but this would be an inconsequential modification. For the components $V_\exists$ and $V_\forall$ we put:

- The $\exists$-vertices $V_\exists \subseteq V$ consist of the pairs of the form $(u, f\sqcup g)$, as well as the pairs (u,a) and $(u,a;h)$ for atomic program a.
- The $\forall$-vertices $V_\forall \subseteq V$ consist of the pairs $(u,f\sqcap g)$, as well as the pairs (X,f) where $(u,X)\in I(a)$ for some atomic action a and state u.

We think of the pairs (u,id) as being *terminal vertices*, and the error vertices are $E\times\{\mathsf{id}\}$.

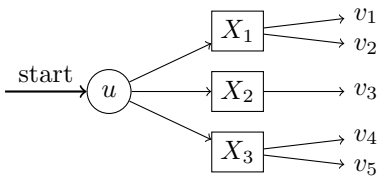
Example 3.8. Suppose that we want to describe a program whose state consists of a single variable x that can take values 0, 1 or 2. The only atomic action that we consider is x++, which assigns $(x+1)$ mod 3 to the variable x. The atomic test $(x=0)$ checks if the value of x is equal to 0. Consider the program

$$h \triangleq \mathsf{while}\,(x=0)\,\mathsf{do}\,((\mathsf{id}\sqcup x{+}{+});(\mathsf{id}\sqcap x{+}{+})).$$

On the right-hand side of Figure 4 we have some abbreviations for parts of the program, and on the left-hand side we see a simplified version of the operational model. We have only drawn the vertices that are reachable from $(0,h)$, $(1,h)$ and $(2,h)$. Since the action x++ is deterministic, we have also made some simplifications such as: the transition sequence $(0,x{+}{+};h)\to(\{1\},\mathsf{id};h)\to(1,h)$ has been reduced to $(0,x{+}{+};h)\to(1,h)$.

The terminal vertices shown in Figure 4 are $(1,\mathsf{id})$ and $(2,\mathsf{id})$. Suppose that $(2,\mathsf{id})$ is the unique error vertex. The winning region $W_\forall$ of the demon consists of:

$$(2,\mathsf{id}) \qquad (2,h) \qquad (1,x{+}{+};h) \qquad (1,g;h) \qquad (0,x{+}{+};g;h)$$



$$\phi = \{(u, X_1), (u, X_2), (u, X_3)\}$$
$$X_1 = \{v_1, v_2\}$$
$$X_2 = \{v_3\}$$
$$X_3 = \{v_4, v_5\}$$

Figure 5: Visualization of a two-round game between the angel and the demon, as described by a relation $\phi \subseteq S \times \wp S$. The angel moves at the circled node, the demon moves at the boxed nodes, and the nodes with no outline are terminal.

The rest of the vertices form the winning region $W_\exists$ of the angel.

4. Denotational Semantics and Full Abstraction

In §3 we presented the syntax of while game schemes and we gave an operational model based on two-player games on finite graphs. Because of this adversarial dynamics, the input-output behavior can no longer be described using binary relations consisting of the possible input-ouput pairs, as is done for usual programs (recall Definition 2.3). Instead, we will adopt an angel-centric view, and we will record in our program denotations the predicates that the angel can guarantee of the output. As usual, a nonempty set S represents the abstract state space, and every test is interpreted as a unary predicate on the state space. Every program term is now interpreted as a binary relation from S to $\wp S$.

Consider such a binary relation $\phi \subseteq S \times \wp S$, which should be thought of as the extension of a dually nondeterministic program. Informally, the pair (u, X) is supposed to belong to ϕ when the following holds: if the program starts at state u, then the angel has a strategy so that whatever the demon does, the final state (supposing that the program terminates) satisfies the predicate X.

The binary relation $\phi \subseteq S \times \wp S$ encodes both the choices of the angel and the demon, and it can be understood intuitively as a two-round game. The angel moves first, and then the demon makes the final move. The options that are available to the angel are given by multiple pairs (u, X_1), (u, X_2), and so on. So, when the game starts at state u, the angel first chooses either X_1, or X_2, or any of the other available options. Suppose that the angel first chooses X_i, where (u, X_i) is in ϕ. Then, during the second round, the demon chooses some final state $v \in X_i$. See Figure 5 for a visualization of this game.

When (u, X) is in ϕ, we understand this as meaning that that the angel can guarantee the predicate X when we start at u. So, it is reasonable to expect that the angel also guarantees from u any predicate that is weaker than X. In order to be consistent with the viewpoint of partial correctness, we also want to require that the angel can guarantee anything in the case of nontermination. Recall Observation 3.4, where we discuss how to summarize two-player games on graphs from the perspective of what the angel can guarantee. These considerations motivate the following definition.

Definition 4.1 (Game Functions)**.** Let S be a nonempty set called the *state space*. We say that $\phi \subseteq S \times \wp S$ is a *game function* on S, denoted $\phi : S \twoheadrightarrow S$, if it satifies:

(1) The set ϕ is *closed upwards*, which is defined to mean the following:

$$(u, X) \in \phi \text{ and } X \subseteq Y \implies (u, Y) \in \phi$$

for every state $u \in S$ and all predicates $X, Y \subseteq S$.

(2) *Non-emptiness*: For every $u \in S$ there is some $X \subseteq S$ with $(u, X) \in \phi$.

Given Condition (1), we can equivalently require that $(u, S) \in \phi$ for every $u \in S$, instead of having Condition (2). This essentially says that the angel always guarantees that the output lies in the state space.

Let $\phi : S \mathrel{\leadsto\!\!\!\!\rightarrow} S$ be a game function. The *options* of the angel at a state $u \in S$, which we denote by $\phi(u)$, is the collection of predicates

$$\phi(u) = \{X \subseteq S \mid (u, X) \in \phi\}.$$

In other words, $\phi(u)$ is the set of all predicates that the angel can guarantee from u. This notation suggests that we can equivalently understand ϕ as being a function $S \to \wp\wp S$. Indeed, the definition says that $(u, X) \in \phi$ iff $X \in \phi(u)$ for all $u \in S$ and $X \subseteq S$.

Now, we will observe that the space of game functions is large enough to encompass nondeterministic functions as a special case. To make this claim precise, we need to define a *lifting* operation, which embeds the nondeterministic functions into the game functions. As we will see, this is not merely an injective map, but it also commutes with the corresponding semantic operations in these two spaces. So, the algebra of nondeterministic functions is embedded via the lifting map into the algebra of game functions.

Definition 4.2 (Lifting & Non-Angelic Game Functions)**.** Let S be a state space, and $k : S \rightsquigarrow S$ be a nondeterministic function on S. We define the *lifting* of k to be the game function $\mathsf{lift}\, k : S \mathrel{\leadsto\!\!\!\!\rightarrow} S$, which is given by

$$\mathsf{lift}\, k \triangleq \{(u, Y) \mid u \in A \text{ and } k(u) \subseteq Y\} : S \mathrel{\leadsto\!\!\!\!\rightarrow} S.$$

This says that for every state $u \in S$ and predicate $Y \subseteq S$: $(u, Y) \in \mathsf{lift}\, k$ iff $k(u) \subseteq Y$. The lifting operation is thus a mapping from the space $S \rightsquigarrow S$ to $S \mathrel{\leadsto\!\!\!\!\rightarrow} S$.

We say that a game function $\phi : S \mathrel{\leadsto\!\!\!\!\rightarrow} S$ is *non-angelic* if it is the lifting of a nondeterministic function, that is, $\phi = \mathsf{lift}\, k$ for some $k : S \rightsquigarrow S$. Essentially, the definition says that the angel always has exactly one minimal choice: for every $u \in S$ there is exactly one minimal predicate $k(u)$ that the angel can guarantee.

Observation 4.3 (Demonic & Angelic Lifting)**.** In Definition 4.2 we consider a lifting operation from the space $S \rightsquigarrow S$ to the space $S \mathrel{\leadsto\!\!\!\!\rightarrow} S$ which interprets the nondeterminism demonically. This works, because a nondeterministic function $k : S \rightsquigarrow S$ records reachability information, i.e. what the demon can achieve. So, we could call lift more descriptively the *demonic lifting* operation. The question then arises of whether we can define an analogous *angelic lifting* operation which interprets the nondeterminism angelically. First, we notice that the space of nondeterministic functions $S \rightsquigarrow S$ with the operations of Figure 2 is inappropriate for modeling pure angelic nondeterminism. Since the angel's goal is safety and the angel wins in the case of nontermination of the program, the semantics should record explicitly when the angel can force divergence. The standard relational semantics of §2, however, is "divergence-oblivious" in the sense of suppressing the information regarding the possibility of divergence. For example, we have that

$$0_S + k = k \text{ for every } k : S \rightsquigarrow S.$$

So, in order to define a reasonable angelic lifting one would have to modify the relational semantics of §2 to record the possibility of nontermination. While this investigation would

Composition **;**	$(u, Z) \in (\phi; \psi) \overset{\text{def}}{\iff}$ there is $Y \subseteq S$ s.t. $(u, Y) \in \phi$, and $(v, Z) \in \psi$ for every $v \in Y$.
Conditional $(\cdot)[-,-]$	$P[\phi, \psi] \triangleq \big(\phi \cap (P \times \wp S)\big) \cup \big(\psi \cap (\sim P \times \wp S)\big)$
	$P[\phi, \psi](u) = \phi(u)$, if $u \in P$
	$P[\phi, \psi](u) = \psi(u)$, if $u \notin P$
Angelic choice $\sqcup$	$\phi \sqcup \psi \triangleq \phi \cup \psi$
Demonic choice $\sqcap$	$\phi \sqcap \psi \triangleq \{(u, X \cup Y) \mid (u, X) \in \phi \text{ and } (u, Y) \in \psi\}$
	$= \phi \cap \psi$
Identity $\mathbf{1}_S$	$\mathbf{1}_S(u) \triangleq \{(u, X) \mid u \in S,\ X \subseteq S \text{ and } u \in X\}$
Zero $\mathbf{0}_S$	$\mathbf{0}_S(u) \triangleq S \times \wp S$
Iteration ($\mathbf{wh} \cdot \mathbf{do} -$)	$\mathbf{wh}\, P\, \mathbf{do}\, \phi \triangleq \bigcap_{\kappa \in \mathbf{Ord}} W_\kappa$, where
	$W_0 \triangleq P[\mathbf{0}_S, \mathbf{1}_S]$
	$W_{\kappa+1} \triangleq P[\phi; W_\kappa, \mathbf{1}_S]$
	$W_\lambda \triangleq \bigcap_{\kappa < \lambda} W_\kappa$, for limit ordinal λ

Figure 6: Semantic operations for game functions.

be interesting mathematically, it is beyond the scope of the present paper. From a practical standpoint, distinguishing the non-angelic game functions (see Definition 4.2) is crucial for the synthesis applications that we consider here. We have to restrict attention to programs where the atomic actions do not involve any angelic choices in order to formulate a reasonable synthesis problem for angelic strategies. Since we are not concerned with the implementation of demonic strategies (the choices of the demon are beyond our control!), the definition of a reasonable angelic lifting operation is of little use here.

We list the formal definitions of the semantic operations on game functions $S \rightsquigarrow S$ in Figure 6. As expected, the angelic choice operation $\sqcup$ increases the options available to the angel. The demonic choice operation $\sqcap$ increases the options of the demon. The identity $\mathbf{1}_S$ is the smallest game function that contains $(u, \{u\})$ for every state $u \in S$. Informally, this definition says that on input u, the angel guarantees output u in the identity game. The intuition for the definition of the zero function $\mathbf{0}_S$ is that when the program diverges, the demon cannot lead the game to an error state, therefore the angel can guarantee anything. This describes a notion of partial correctness.

Example 4.4. We will calculate now the denotation of the program h from Example 3.8. We write S for the state space, and I for the interpretation of the atomic symbols. We present below a table with the denotations of all subprograms of h.

$$p = (x = 0) \qquad f = \mathsf{id} \sqcup x\mathsf{++} \qquad g = \mathsf{id} \sqcap x\mathsf{++} \qquad h = \mathsf{W}p(f; g)$$

Since the options of the angel are closed upwards, it suffices to record the minimal predicates for every state. Define $P = I(p) = \{0\}$, and we have:

state	$\mathbf{0}_S$	$\mathbf{1}_S$	$I(x{+}{+})$	$\phi = I(f)$	$\psi = I(g)$	$\phi;\psi = I(f;g)$	W_0	$\phi;\psi;W_0$	W_1
0	$\emptyset$	$\{0\}$	$\{1\}$	$\{0\}$ $\{1\}$	$\{0,1\}$	$\{0,1\}$ $\{1,2\}$	$\emptyset$	$\{1\}$ $\{1,2\}$	$\{1\}$
1	$\emptyset$	$\{1\}$	$\{2\}$	$\{1\}$ $\{2\}$	$\{1,2\}$	$\{1,2\}$ $\{2,0\}$	$\{1\}$	$\{1,2\}$ $\{2\}$	$\{1\}$
2	$\emptyset$	$\{2\}$	$\{0\}$	$\{2\}$ $\{0\}$	$\{2,0\}$	$\{2,0\}$ $\{0,1\}$	$\{2\}$	$\{2\}$ $\{1\}$	$\{2\}$

where $W_0 = P\big[\mathbf{0}_S, \mathbf{1}_S\big]$ and $W_1 = P\big[\phi;\psi;W_0, \mathbf{1}_S\big]$. We leave as an exercise to the reader to verify that $W_2 = P\big[\phi;\psi;W_1, \mathbf{1}_S\big] = W_1$. It follows that $I(h) = W_2$.

We note that the definition of Figure 6 gives the while operation as a greatest fixpoint. This is not surprising, because the semantics we consider is meant to be useful for reasoning about *safety properties*. As we will see, this definition agrees with the standard least fixpoint definition of while loops when there is only one kind of nondeterminism (Lemma 4.5 below). More importantly, we will prove that our definition is *exactly correct*, becauses it agrees with the intended operational semantics of dual nondeterminism (Theorem 4.11).

Lemma 4.5 (Lifting Commutes With The Semantic Operations)**.** Let k and ℓ be nondeterministic functions on S, and P be a unary predicate on S. Then, the following hold:

$$\mathsf{lift}\, 0_S = \mathbf{0}_S \qquad \mathsf{lift}(k;\ell) = (\mathsf{lift}\, k);(\mathsf{lift}\, \ell) \qquad \mathsf{lift}(P[k,\ell]) = P\big[\mathsf{lift}\, k, \mathsf{lift}\, \ell\big]$$
$$\mathsf{lift}\, 1_S = \mathbf{1}_S \qquad \mathsf{lift}(k+\ell) = (\mathsf{lift}\, k) \sqcap (\mathsf{lift}\, \ell) \qquad \mathsf{lift}(\mathrm{wh}\, P\, \mathrm{do}\, k) = \mathbf{wh}\, P\, \mathbf{do}\, (\mathsf{lift}\, k)$$

So, the lifting map commutes with all the semantic operations of nondeterministic functions.

Proof. The cases of 0, 1, demonic choice and conditionals are straightforward and we omit them. For the case of composition we have that:

$(u,Z) \in \mathsf{lift}(k;\ell) \iff$	[def. of lift]
$(k;\ell)(u) \subseteq Z \iff$	[def. of ;]
$\bigcup_{v\in k(u)} \ell(v) \subseteq Z \iff$	[union and $\subseteq$]
$\ell(v) \subseteq Z$ for every $v \in k(u) \iff$	[for "$\Rightarrow$" put $Y = k(u)$]
$\exists Y \subseteq S.\ \ k(u) \subseteq Y$ and $\ell(v) \subseteq Z$ for all $v \in Y \iff$	[def. of lift]
$\exists Y \subseteq S.\ \ (u,Y) \in \mathsf{lift}\, k$ and $(v,Z) \in \mathsf{lift}\, \ell$ for all $v \in Y \iff$	[def. of **;**]
$(u,Z) \in (\mathsf{lift}\, k);(\mathsf{lift}\, \ell)$.	

Since $u \in S$ and $Z \subseteq S$ above are arbitrary, we have established $\mathsf{lift}(k;\ell) = (\mathsf{lift}\, k);(\mathsf{lift}\, \ell)$. It remains to consider the case of $\mathrm{wh}\, P\, \mathrm{do}\, k$. We put $\phi = \mathsf{lift}\, k : S \nrightarrow S$, and we recall the definitions for the semantic iteration operations:

$$\mathrm{wh}\, P\, \mathrm{do}\, k = \textstyle\sum_{\kappa\in\mathbf{Ord}} V_n \qquad \mathbf{wh}\, P\, \mathbf{do}\, \phi = \bigcap_{\kappa\in\mathbf{Ord}} W_\kappa$$
$$V_0 = P[0_S, 1_S] \qquad W_0 = P\big[\mathbf{0}_S, \mathbf{1}_S\big]$$
$$V_{\kappa+1} = P[k; V_\kappa, 1_S] \qquad W_{\kappa+1} = P\big[\phi; W_\kappa, \mathbf{1}_S\big]$$
$$V_\lambda = \textstyle\sum_{\kappa<\lambda} V_\kappa,\ \text{limit ordinal } \lambda \qquad W_\lambda = \bigcap_{\kappa<\lambda} W_\kappa,\ \text{limit ordinal } \lambda$$

It is a well-known fact that $\mathrm{wh}\, P\, \mathrm{do}\, k = V_\omega$, which says that the least fixpoint closes at ω iterations. The crucial observation now is that

$$W_\kappa = \mathsf{lift}\, V_\kappa \text{ for every ordinal } \kappa.$$

This is shown by transfinite induction on ordinals. The proof involves using the commutation results for lift (for 0, 1, conditionals, composition) that we have shown so far. Finally,

$$\begin{aligned}(u,Y)\in\mathsf{lift}(\mathrm{wh}\,P\,\mathrm{do}\,k) &\iff (\mathrm{wh}\,P\,\mathrm{do}\,k)(u)\subseteq Y\\ &\iff (\textstyle\sum_\kappa V_\kappa)(u)=\bigcup_\kappa V_\kappa(u)\subseteq Y\\ &\iff V_\kappa(u)\subseteq Y \text{ for every ordinal } \kappa\\ &\iff (u,Y)\in \mathsf{lift}\,V_\kappa = W_\kappa \text{ for every ordinal } \kappa\\ &\iff (u,Y)\in\textstyle\bigcap_\kappa W_\kappa = \mathbf{wh}\,P\,\mathbf{do}\,\phi.\end{aligned}$$

We have thus shown that $\mathsf{lift}(\mathrm{wh}\,P\,\mathrm{do}\,k)=\mathbf{wh}\,P\,\mathbf{do}\,(\mathsf{lift}\,k)$ and the proof is complete. □

Essentially, the above lemma says that the game function operations are a generalization of the nondeterministic function operations. It is an easy exercise to show that the map lift is injective. So, the algebra $S\rightsquigarrow S$ with the operations of Figure 2 is embedded via lift into the algebra $S\twoheadrightarrow S$ with the operations of Figure 6.

Definition 4.6 (The Implementation Relation)**.** Let $k : S\rightsquigarrow S$ be a nondeterministic function and $\phi : S\twoheadrightarrow S$ be a game function. We say that k *implements* ϕ if $\mathsf{lift}\,k\subseteq f$, and we denote this by $k\sqsubseteq\phi$. The definition is meant to capture the idea that k resolves (in some possible way) the angelic nondeterminism of ϕ. To put it differently, the function k *chooses* for every start state u an output predicate $k(u)\in\phi(u)$ that the angel can guarantee.

Lemma 4.7 (The Implementation Calculus)**.** The relation $\sqsubseteq$ satisfies the following rules:

$$1_A\sqsubseteq\mathbf{1}_A \qquad 0_{AB}\sqsubseteq\mathbf{0}_{AB} \qquad \frac{P\subseteq S \qquad k\sqsubseteq\phi \qquad \ell\sqsubseteq\psi}{P[k,\ell]\sqsubseteq P\big[\phi,\psi\big]} \qquad \frac{k\sqsubseteq\phi \qquad \ell\sqsubseteq\psi}{k;\ell\sqsubseteq\phi;\psi}$$

$$\frac{k\sqsubseteq\phi}{k\sqsubseteq\phi\sqcup\psi} \qquad \frac{\ell\sqsubseteq\psi}{\ell\sqsubseteq\phi\sqcup\psi} \qquad \frac{k\sqsubseteq\phi \quad \ell\sqsubseteq\psi}{k+\ell\sqsubseteq\phi\sqcap\psi} \qquad \frac{P\subseteq S \qquad k\sqsubseteq\phi}{\mathrm{wh}\,P\,\mathrm{do}\,k\sqsubseteq\mathbf{wh}\,P\,\mathbf{do}\,\phi}$$

where $k,\ell : S\rightsquigarrow S$ are nondeterministic functions and $\phi,\psi : S\twoheadrightarrow S$ are game functions.

Proof. First, we note that all the operations on game functions are monotone w.r.t. inclusion. That is, if $\phi\subseteq\phi'$ and $\psi\subseteq\psi'$ then we also have:

$$\phi;\psi\subseteq\phi';\psi' \qquad \phi\sqcup\psi\subseteq\phi'\sqcup\psi' \qquad \mathbf{wh}\,P\,\mathbf{do}\,\phi\subseteq\mathbf{wh}\,P\,\mathbf{do}\,\phi'$$
$$P\big[\phi,\psi\big]\subseteq P\big[\phi',\psi'\big] \qquad \phi\sqcap\psi\subseteq\phi'\sqcap\psi'$$

Assume now that $k\sqsubseteq\phi$ and $\ell\sqsubseteq\psi$, i.e., $\mathsf{lift}\,k\subseteq\phi$ and $\mathsf{lift}\,\ell\subseteq\psi$. We obtain the inclusions

$$\begin{aligned}\mathsf{lift}\,1_S=\mathbf{1}_S\subseteq\mathbf{1}_S &\qquad \mathsf{lift}(k;\ell)=(\mathsf{lift}\,k);(\mathsf{lift}\,\ell)\subseteq\phi;\psi\\ \mathsf{lift}\,0_S=\mathbf{0}_S\subseteq\mathbf{0}_S &\qquad \mathsf{lift}(P[k,\ell])=P\big[\,\mathsf{lift}\,k,\mathsf{lift}\,\ell\big]\subseteq P\big[\phi,\psi\big]\\ \mathsf{lift}\,k\subseteq\phi\subseteq\phi\cup\psi=\phi\sqcup\psi &\qquad \mathsf{lift}(\mathrm{wh}\,P\,\mathrm{do}\,k)=\mathbf{wh}\,P\,\mathbf{do}\,(\mathsf{lift}\,k)\subseteq\mathbf{wh}\,P\,\mathbf{do}\,\phi\\ \mathsf{lift}\,\ell\subseteq\psi\subseteq\phi\cup\psi=\phi\sqcup\psi &\qquad \mathsf{lift}(k+\ell)=(\mathsf{lift}\,k)\sqcap(\mathsf{lift}\,\ell)\subseteq\phi\sqcap\psi\end{aligned}$$

using the monotonicity properties for game function operations and the fact that the lifting operation commutes with the semantic program operations (Lemma 4.5). □

Definition 4.8 (Game Interpretation)**.** As in the case of nondeterministic program schemes (Definition 2.3), an interpretation of the language of while game schemes consists of a nonempty *state space* S and an *interpretation function* I. For a program term f, its *interpretation* $I(f) : S \rightsquigarrow S$ is a game function on S. The function I specifies the meaning of every atomic test, and extends to all tests in the obvious way. Moreover, I specifies the meaning $I(a) : S \rightsquigarrow S$ of every atomic action. It extends to all game schemes as:

$$I(\mathsf{id}) = \mathbf{1}_S \quad I(f;g) = I(f);I(g) \quad I(f \sqcup g) = I(f) \sqcup I(g) \quad I(p[f,g]) = I(p)\big[I(f), I(g)\big]$$
$$I(\bot) = \mathbf{0}_S \qquad\qquad I(f \sqcap g) = I(f) \sqcap I(g) \qquad I(\mathsf{W}pf) = \mathbf{wh}\, I(p)\, \mathbf{do}\, I(f)$$

We say that the game interpretation I *lifts* the nondeterministic interpretation R if they have the same state space, and additionally:

(i) $I(p) = R(p)$ for every atomic test p, and
(ii) $I(a) = \mathsf{lift}\, R(a)$ for every atomic program a.

We also say that I *is the lifting of* R.

Definition 4.9 (Chain Property)**.** A decreasing chain of predicates is a transfinite sequence $(X_\kappa)_{\kappa \in \mathbf{Ord}}$ with $X_\kappa \supseteq X_\lambda$ for ordinals $\kappa \leq \lambda$. Let $\phi : S \rightsquigarrow S$ be a game function. We say that ϕ satisfies the *chain property* if for every state $u \in S$ and every decreasing chain $(Y_\kappa)_\kappa$ of predicates on S, $(u, Y_\kappa) \in \phi$ for all κ implies that $(u, \bigcap_\kappa Y_\kappa) \in \phi$.

Lemma 4.10 (Preservation of Chain Property)**.** The following hold:

(1) Every non-angelic game function satisfies the chain property.
(2) The game functions $\mathbf{0}_S$ and $\mathbf{1}_S$ satisfy the chain property.
(3) If the game functions $\phi, \psi : S \rightsquigarrow S$ satisfy the chain property, then so do the game functions $P\big[\phi, \psi\big]$, $\phi;\psi$, $\phi \sqcup \psi$, $\phi \sqcap \psi$, and $\mathbf{wh}\, P\, \mathbf{do}\, \phi$, where P is a predicate on S.

Proof. The most interesting parts of the proof are showing that the operations of angelic choice and composition preserve the chain property. We omit the rest of the proof, since the reader can easily reconstruct it.

For the case $\phi \sqcup \psi$ of angelic choice, assume that $(u, Y_\kappa) \in \phi \sqcup \psi$ for every ordinal κ. We recall the definition $\phi \sqcup \psi = \phi \cup \psi$, which means that $(u, Y_\kappa) \in \phi$ or $(u, Y_\kappa) \in \psi$ for all κ. Define the classes $O(\phi)$ and $O(\psi)$ of ordinals as follows:

$$O(\phi) = \{\lambda \in \mathbf{Ord} \mid (u, Y_\lambda) \in \phi\} \qquad O(\psi) = \{\mu \in \mathbf{Ord} \mid (u, Y_\mu) \in \psi\}$$

Clearly, the equality $O(\phi) \cup O(\psi) = \mathbf{Ord}$ holds. This implies that at least one of the classes $O(\phi)$, $O(\psi)$ has no upper bound. By symmetry, we only consider the case where $O(\phi)$ has no upper bound, that is: for every ordinal κ there is some $\lambda \geq \kappa$ with $\lambda \in O(\phi)$. We extend the subsequence $(Y_\lambda)_{\lambda \in O(\phi)}$ into a decreasing chain $(\hat{Y}_\lambda)_{\lambda \in \mathbf{Ord}}$ as:

$$\hat{Y}_\lambda = Y_{\lambda'}, \text{ where } \lambda' = \mathrm{least}\{\kappa \in \mathbf{Ord} \mid \kappa \geq \lambda \text{ and } \kappa \in O(\phi)\}.$$

In particular, if $\lambda \in O(\phi)$ then $\hat{Y}_\lambda = Y_\lambda$. It is straightforward to verify that $(\hat{Y}_\lambda)_{\lambda \in \mathbf{Ord}}$ is a decreasing chain with $(u, \hat{Y}_\lambda) \in \phi$ for every $\lambda \in \mathbf{Ord}$. Since ϕ satisfies the chain property, we get that $(u, \bigcap_{\lambda \in \mathbf{Ord}} \hat{Y}_\lambda) \in \phi$. Finally, we observe that

$$\textstyle\bigcap_{\kappa \in \mathbf{Ord}} Y_\kappa = \bigcap_{\lambda \in O(f)} Y_\lambda = \bigcap_{\lambda \in \mathbf{Ord}} \hat{Y}_\lambda.$$

This gives us the desired $(u, \bigcap_{\kappa \in \mathbf{Ord}} Y_\kappa) \in \phi \subseteq \phi \cup \psi$. So, $\phi \sqcup \psi$ satisfies the chain property.

For the case $\phi;\psi$ of composition, we consider the decreasing chain $(Z_\kappa)_\kappa$ and we assume that $(u, Z_\kappa) \in (\phi;\psi)$ for all κ. For every ordinal κ, define the collection of predicates

$$\mathcal{Y}_\kappa = \{Y \subseteq S \mid (u, Y) \in \phi \text{ and } (v, Z_\kappa) \in \psi \text{ for all } v \in Y.\}$$

The assumption $(u, Z_\kappa) \in (\phi;\psi)$ means that the collection $\mathcal{Y}_\kappa$ is nonempty. We then define the predicate $Y_\kappa = \bigcup \mathcal{Y}_\kappa$ and we observe that $Y_\kappa \in \mathcal{Y}_\kappa$, that is:

$$(u, Y_\kappa) \in \phi \qquad \text{and} \qquad (v, Z_\kappa) \in \psi \text{ for all } v \in Y_\kappa.$$

Moreover, the implications $\kappa \leq \lambda \Rightarrow Z_\kappa \supseteq Z_\lambda \Rightarrow \mathcal{Y}_\kappa \supseteq \mathcal{Y}_\lambda \Rightarrow Y_\kappa \supseteq Y_\lambda$ hold. This means that the sequence $(Y_\kappa)_\kappa$ is a decreasing chain. The third containment is justified as follows:

$$\begin{aligned} Y \in \mathcal{Y}_\lambda &\implies (u, Y) \in \phi \text{ and } (v, Z_\lambda) \in \psi \text{ for all } v \in Y \\ &\implies (u, Y) \in \phi \text{ and } (v, Z_\kappa) \in \psi \text{ for all } v \in Y \\ &\implies Y \in \mathcal{Y}_\kappa. \end{aligned}$$

Since ϕ satisfies the chain property, we obtain that $(u, \bigcap_\kappa Y_\kappa) \in \phi$. Let us consider now an arbitrary element v of $\bigcap_\kappa Y_\kappa$. We get that $v \in Y_\kappa$ and hence $(v, Z_\kappa) \in \psi$ for every ordinal κ. But ψ also satisfies the chain property, which gives us that $(v, \bigcap_\kappa Z_\kappa) \in \psi$. We know that:

$$(u, \textstyle\bigcap_\kappa Y_\kappa) \in \phi \qquad \text{and} \qquad (v, \bigcap_\kappa Z_\kappa) \in \psi \text{ for all } v \in \bigcap_\kappa Y_\kappa.$$

This means that $(u, \bigcap_\kappa Z_\kappa) \in (\phi;\psi)$. We conclude that $\phi;\psi$ satisfies the chain property. □

Theorem 4.11 (Full Abstraction)**.** *Let I be an interpretation of atomic tests as unary predicates on a state space S and of atomic actions as game functions $S \rightsquigarrow S$ that satisfy the chain property. Then, for every while game scheme f, state $u \in S$ and predicate $Y \subseteq S$ we have that: $(u, Y) \in I(f)$ iff Player $\exists$ (the angel) has a winning strategy from the vertex (u, f) in the safety game $G_I(f, \sim Y)$ (recall Definition 3.7).*

Proof. The proof is by induction on the structure of f.

First, we consider the case of the atomic action a. Recall that we have $C(a) = \{a, \mathsf{id}\}$. The start vertex for the game is (u, a). The angel has a winning strategy from (u, a) iff there exists some predicate X such that $(u, X) \in I(a)$ and $X \subseteq Y$.

$$(u, a) \to (X, \mathsf{id}) \to (v, \mathsf{id}), \text{ where } v \in X$$

For the case of the skip program id, we have that $C(\mathsf{id}) = \{\mathsf{id}\}$. The start vertex for the game is (u, id), and it is also a terminal vertex. So, the angel has a winning strategy in the game $G_I(\mathsf{id}, \sim Y)$ iff $u \in Y$ iff $(u, Y) \in I(\mathsf{id}) = \mathbf{1}_S$.

We handle now the case of the conditional $p[f, g]$. We have that $C(p[f, g]) = \{p[f, g]\} \cup C(f) \cup C(g)$. Consider a pair (u, Y), where $u \in I(p)$. The case where $u \in I(\neg p)$ is analogous, and we omit it. Notice that there exists a unique transition $(u, p[f, g]) \to (u, f)$. This means that after the transition is taken, any play in $G_I(p[f, g], \sim Y)$ is the same as a play in the game $G_I(f, \sim Y)$. So, we obtain the equivalences:

$$\begin{aligned} &(u, X) \in I(p[f, g]) \iff (u, X) \in I(f) \iff \\ &\text{The angel has a winning strategy from } (u, f) \text{ in } G_I(f, \sim Y) \iff \\ &\text{The angel has a winning strategy from } (u, p[f, g]) \text{ in } G_I(p[f, g], \sim Y). \end{aligned}$$

The cases $f \sqcup g$ and $f \sqcap g$ are handled using similar arguments to the ones we used for the conditional $p[f, g]$, and we therefore omit them.

We will prove now the claim for the while loop $\mathsf{W}pf$. Recall that $C(\mathsf{W}pf) = \{\mathsf{W}pf, \mathsf{id}\} \cup C(f)@\mathsf{W}pf$ and $I(\mathsf{W}pf) = \bigcap_{\kappa \in \mathbf{Ord}} W_\kappa$, where the transfinite sequence W_κ is given by

$$W_0 \triangleq I(p)[\mathbf{0}_S, \mathbf{1}_S] \quad W_{\kappa+1} \triangleq I(p)[I(f); W_\kappa, \mathbf{1}_S] \quad W_\lambda \triangleq \bigcap_{\kappa<\lambda} W_\kappa, \text{ limit ordinal } \lambda$$

Consider the predicate $Y \subseteq S$, and define the transfinite sequence $(X_\kappa)_{\kappa \in \mathbf{Ord}}$ as follows:

$$\begin{aligned} X_0 &= I(p) \cup (\sim I(p) \cap Y) \\ X_{\kappa+1} &= \{u \in S \mid u \in I(p) \text{ and } (u, X_\kappa) \in I(f)\} \cup (\sim I(p) \cap Y) \\ X_\lambda &= \textstyle\bigcap_{\kappa<\lambda} X_\kappa, \text{ for limit ordinal } \lambda \end{aligned}$$

The sequence $(X_\kappa)_\kappa$ can be defined equivalently in terms of the approximants W_κ, as the claim below states. We also put $X = \bigcap_{\kappa \in \mathbf{Ord}} X_\kappa$. A transfinite induction on κ establishes:

Claim. $X_\kappa = \{u \in S \mid (u, Y) \in W_\kappa\}$ for every ordinal κ. □

The above claim implies in particular that

$$X = \{u \in S \mid (u, Y) \in I(\mathsf{W}pf)\}.$$

Moreover, we see below that X is an "inductive invariant" for the while loop $\mathsf{W}pf$.

Claim. If $u \in I(p)$ and $u \in X$, then (u, X) is in $I(f)$.

Proof. Suppose that $u \in I(p)$ and $u \in X$, which implies that $u \in X_{\kappa+1}$ for every κ. From the inductive definition of X_κ, we obtain that $(u, X_\kappa) \in I(f)$ for every κ. Since every interpretation $I(a)$ for atomic action a satisfies the chain property, we obtain from Lemma 4.10 that $I(f)$ satisfies the chain property. It follows that $(u, X) \in I(f)$. □

Let us consider now the game $G_I(\mathsf{W}pf, \sim Y)$.

- Consider a state $u \in I(p)$ with $u \in X$. The previous claim says that $(u, X) \in I(f)$, and hence the I.H. gives us that the angel has a winning strategy σ_u in the game $G_I(f, \sim X)$. We define the $\exists$-strategy σ in the game $G_I(\mathsf{W}pf, \sim Y)$ as follows: every time a vertex $(u, \mathsf{W}pf)$ with $u \in I(p)$ is encountered, start playing according to σ_u. Notice that we have the transition $(u, \mathsf{W}pf) \to (u, f@\mathsf{W}pf)$, which means that σ simulates σ_u on $G_I(f, \sim X)$.

  It follows that when the angel plays according to σ in the game $G_I(\mathsf{W}pf, \sim Y)$ with start vertex $(u, \mathsf{W}pf)$ where $u \in X$, the play will never hit an error vertex in $\sim Y \times \{\mathsf{id}\}$. In particular, if $(u, Y) \in I(\mathsf{W}pf)$ then $u \in X$ and hence the angel has a winning strategy from $(u, \mathsf{W}pf)$ in the game $G_I(\mathsf{W}pf, \sim Y)$.
- Let U be the set of states $u \in S$ for which the angel has a winning strategy from $(u, \mathsf{W}pf)$ in the game $G_I(\mathsf{W}pf, \sim Y)$. Let σ be the (w.l.o.g. memoryless, see Theorem 3.3) strategy of Player $\exists$ that witnesses his winning region in the game $G_I(\mathsf{W}pf, \sim Y)$.

  Consider a state $u \in I(p)$ with $u \in U$. If the angel plays according to σ in the game $G_I(f, \sim U)$, then he wins, because σ keeps the play within the winning region. The I.H. then says that $(u, U) \in I(f)$.

  Claim 4.12. $U \subseteq X$.

  Proof. It suffices to show that $U \subseteq X_\kappa$ for every ordinal κ. For the base case $\kappa = 0$, the claim $U \subseteq X_0 = I(p) \cup (\sim I(p) \cap Y)$ is obvious. For successor ordinals:

$$\begin{aligned} X_{\kappa+1} &= \{u \in S \mid u \in I(p) \text{ and } (u, X_\kappa) \in I(f)\} \cup (\sim I(p) \cap Y) \\ &\supseteq \{u \in S \mid u \in I(p) \text{ and } (u, U) \in I(f)\} \cup (\sim I(p) \cap Y) \\ &\supseteq \{u \in S \mid u \in I(p) \text{ and } u \in U\} \cup (\sim I(p) \cap Y), \end{aligned}$$

  which is equal to U. The case of limit ordinals is easy. □

Suppose now that $(u, \mathsf{W}pf)$ is in the winning region of the angel in the game $G_I(\mathsf{W}pf, {\sim}Y)$. It follows that $u \in U$ and hence $u \in X$. We thus conclude that (u, Y) is in $I(\mathsf{W}pf)$.

This completes the proof for the case of the while loop $\mathsf{W}pf$.

Finally, we have to deal with the case $e; f$ of sequential composition. Recall the definitions $C(e; f) = C(e)@f \cup C(f)$ and $I(e; f) = I(e)\mathbf{;}\, I(f)$.

- Suppose that $(u, Z) \in I(e; f)$. There exists $Y \subseteq S$ with $(u, Y) \in I(e)$ and $(v, Z) \in I(f)$ for every $v \in Y$. The I.H. says that there exists a winning $\exists$-strategy σ for the game $G_I(e, {\sim}Y)$ started at vertex (u, e). Moreover, for every $v \in Y$, there exists a winning $\exists$-strategy τ_v for the game $G_I(f, {\sim}Z)$ started at vertex (v, f). Now, we define the strategy ρ for the game $G_I(e; f, {\sim}Y)$ as follows: start playing according to σ, and as soon as you encounter a vertex (v, f) start playing according to τ_v. The $\exists$-strategy ρ is winning for the angel in the game $G_I(e; f, {\sim}Z)$ when started at $(u, e; f)$.
- Suppose now that the angel has a (w.l.o.g. memoryless, see Theorem 3.3) winning strategy ρ from the vertex $(u, e; f)$ in the game $G_I(e; f, {\sim}Z)$. Let

$$Y = \{v \in S \mid \text{the vertex } (v, \mathsf{id}; f) \text{ appears in some } \rho\text{-play starting from } (u, e; f)\}.$$

  Then, the angel has a winning strategy from (u, e) in the game $G_I(e, {\sim}Y)$. Moreover, for every $v \in Y$, the angel has a winning strategy from (v, f) in the game $G_I(f, {\sim}Z)$. From the I.H., it follows that $(u, Y) \in I(e)$. Moreover, for every $v \in Y$, we obtain that $(v, Z) \in I(f)$. So, $(u, Z) \in I(e; f)$.

This concludes the argument for the case of composition, and the proof is thus complete. □

5. A Hoare Calculus for While Game Schemes

In this section, we present formulas that are used to specify programs. The basic formulas are Hoare assertions of the form $\{p\}f\{q\}$, and we also consider assertions under certain hypotheses Φ, Ψ of a simple form. The latter formulas are called Hoare implications and are of the form $\Phi, \Psi \Rightarrow \{p\}f\{q\}$. We will then continue to present our first axiomatization, with which we derive valid Hoare implications.

Definition 5.1 (Tests and Entailment)**.** Let I be an interpretation of the atomic tests, which extends to all tests in the obvious way. For a test p and a state $u \in S$, we write $I, u \models p$ when $u \in I(p)$. We read this as: "the state u satisfies p (under I)". When $I, u \models p$ for every state $u \in S$, we say that I *satisfies* p, and we write $I \models p$. For a set Φ of tests, the interpretation I *satisfies* Φ if it satisfies every test in Φ. We then write $I \models \Phi$. Finally, we say that Φ *entails* p, denoted $\Phi \models p$, if $I \models \Phi$ implies $I \models p$ for every I.

Definition 5.2 (Hoare Assertions)**.** An expression $\{p\}f\{q\}$, where p and q are tests and f is a program term, is called a *Hoare assertion*. The test p is called the *precondition* and the test q is called the *postcondition* of the assertion. Informally, the formula $\{p\}f\{q\}$ says that when the program f starts at a state satisfying the predicate p, then the angel has a strategy so that whatever the demon does, the final state (upon termination) satisfies the predicate q. The Hoare assertion $\{p\}a\{q\}$, where a is an atomic program, is called a *simple Hoare assertion*. More formally, we say that the interpretation I *satisfies* $\{p\}f\{q\}$ when

$$I, u \models p \text{ implies that } (u, I(q)) \in I(f)$$

for every state $u \in S$. We then write $I \models \{p\}f\{q\}$.

Definition 5.3 (Simple Hoare Implications & Weak Hoare Theory)**.** Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. We call the expression

$$\Phi, \Psi \Rightarrow \{p\}f\{q\}$$

a *simple Hoare implication*. The tests in Φ and the simple assertions in Ψ are the *hypotheses* of the implication, and the Hoare assertion $\{p\}f\{q\}$ is the *conclusion*. We use the qualifier *simple* for implications of the form $\Phi, \Psi \Rightarrow \{p\}f\{q\}$, because the hypotheses Ψ involve only simple Hoare assertions (instead of general Hoare assertions for arbitrary programs).

Let I be an interpretation of tests and actions. We say that I *satisfies* the implication $\Phi, \Psi \Rightarrow \{p\}f\{q\}$, which we denote by $I \models \Phi, \Psi \Rightarrow \{p\}f\{q\}$, when the following holds: If the interpretation I satisfies every test in Φ and every assertion in Ψ, then I satisfies the assertion $\{p\}f\{q\}$. An implication $\Phi, \Psi \Rightarrow \{p\}f\{q\}$ is *valid*, denoted $\Phi, \Psi \models \{p\}f\{q\}$, if every interpretation satisfies it. The set of all valid Hoare implications forms the *weak Hoare theory* of while game schemes.

Definition 5.4 (Boolean Atoms & Φ-Consistency)**.** Suppose that we have fixed a finite set of atomic tests. For an atomic test p, the expressions p and $\neg p$ are called *literals* for p (*positive* and *negative* respectively). Fix an enumeration $p_1, p_2, \ldots, p_k$ of the atomic tests. A *Boolean atom* (or simply *atom*) is an expression $\ell_1\ell_2\cdots\ell_k$, where every ℓ_i is a literal for p_i. We use lowercase letters $\alpha, \beta, \gamma, \ldots$ from the beginning of the Greek alphabet to range over atoms. An atom is essentially a conjunction of literals, and it can also be thought of as a propositional truth assignment. We write $\alpha \leq p$ to mean that the atom α satisfies the test p. We denote by At the set of all atoms.

Assume that Φ is a finite set of tests. We say that an atom α is Φ-*consistent* if $\alpha \leq p$ for every test p in Φ. We write At_Φ for the set of all Φ-consistent atoms.

Definition 5.5 (The Free Test Interpretation)**.** Let Φ be a finite set of tests. We define the interpretation I_Φ on tests, which is called the *free test interpretation* w.r.t. Φ. The state space is the set At_Φ of Φ-consistent atoms, and every test is interpreted as a unary predicate on At_Φ. For an atomic test p, define its interpretation

$$I_\Phi(p) \triangleq \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p\}$$

to be the set of Φ-consistent atoms that satisfy p. In fact, an easy induction on the structure of tests proves that for every (atomic or composite) test p, $I_\Phi(p)$ is equal to the set of Φ-consistent atoms that satisfy p.

Note 5.6 (Complete Boolean Calculus)**.** We assume that we have a complete Boolean calculus, with which we derive judgments $\Phi \vdash p$, where Φ is a finite set of tests and p is a test. This means that the statements

$$\Phi \models p \qquad I_\Phi \models p \qquad I_\Phi(p) = \mathsf{At}_\Phi \qquad \Phi \vdash p$$

are all equivalent. From this we also obtain that $I_\Phi(p) \subseteq I_\Phi(q)$ iff $\Phi \vdash p \to q$.

We propose now a Hoare-style calculus (Figure 7), which is used for deriving simple Hoare implications that involve while game schemes. As we will show, the calculus of Figure 7 is sound and complete for the weak Hoare theory of while game schemes. Establishing soundness is a relatively straightforward result. The most interesting part is the soundness of the (loop) rule for while loops. The observation is that the loop invariant defines a "safe region" of the game, and the angel has a strategy to keep a play within this region.

$$\frac{\{p\}a\{q\} \text{ in } \Psi}{\Phi, \Psi \vdash \{p\}a\{q\}}\,(\mathsf{hyp}) \qquad \frac{}{\Phi, \Psi \vdash \{p\}\mathsf{id}\{p\}}\,(\mathsf{skip}) \qquad \frac{}{\Phi, \Psi \vdash \{p\}\bot\{q\}}\,(\mathsf{dvrg})$$

$$\frac{\begin{array}{c}\Phi, \Psi \vdash \{p\}f\{q\} \\ \Phi, \Psi \vdash \{q\}g\{r\}\end{array}}{\Phi, \Psi \vdash \{p\}f; g\{r\}}\,(\mathsf{seq}) \qquad \frac{\begin{array}{c}\Phi, \Psi \vdash \{q \wedge p\}f\{r\} \\ \Phi, \Psi \vdash \{q \wedge \neg p\}g\{r\}\end{array}}{\Phi, \Psi \vdash \{q\}\mathsf{if}\, p\, \mathsf{then}\, f\, \mathsf{else}\, g\{r\}}\,(\mathsf{cond})$$

$$\frac{\Phi, \Psi \vdash \{r \wedge p\}f\{r\}}{\Phi, \Psi \vdash \{r\}\mathsf{while}\, p\, \mathsf{do}\, f\{r \wedge \neg p\}}\,(\mathsf{loop})$$

$$\frac{\Phi, \Psi \vdash \{p\}f_i\{q\}}{\Phi, \Psi \vdash \{p\}f_1 \sqcup f_2\{q\}}\,(\mathsf{ang}_i) \qquad \frac{\Phi, \Psi \vdash \{p\}f\{q\} \qquad \Phi, \Psi \vdash \{p\}g\{q\}}{\Phi, \Psi \vdash \{p\}f \sqcap g\{q\}}\,(\mathsf{dem})$$

$$\frac{\Phi \vdash p' \to p \qquad \Phi, \Psi \vdash \{p\}f\{q\} \qquad \Phi \vdash q \to q'}{\Phi, \Psi \vdash \{p'\}f\{q'\}}\,(\mathsf{weak})$$

$$\frac{\Phi, \Psi \vdash \{p_1\}f\{q\} \qquad \Phi, \Psi \vdash \{p_2\}f\{q\}}{\Phi, \Psi \vdash \{p_1 \vee p_2\}f\{q\}}\,(\mathsf{join}) \qquad \begin{array}{l}\Phi, \Psi \vdash \{\mathsf{false}\}f\{q\} \quad (\mathsf{join}_0) \\ \Phi, \Psi \vdash \{p\}f\{\mathsf{true}\} \quad (\mathsf{meet}_0)\end{array}$$

Figure 7: *Game Hoare Logic*: A sound and complete Hoare-style calculus for while program schemes with angelic and demonic nondeterministic choice.

Observation 5.7 (Variant Rule for Demonic Choice)**.** We can have a slightly more flexible form of the rule for demonic choice. The following rule is admissible:

$$\frac{\Phi, \Psi \vdash \{p\}f\{q\} \qquad \Phi, \Psi \vdash \{p\}g\{r\}}{\Phi, \Psi \vdash \{p\}f \sqcap g\{q \vee r\}}\,(\mathsf{dem}').$$

The proof that (dem') is admissible is straightforward:

$$\frac{\dfrac{\Phi, \Psi \vdash \{p\}f\{q\} \qquad \Phi \vdash q \to q \vee r}{\Phi, \Psi \vdash \{p\}f\{q \vee r\}}\,(\mathsf{weak}) \qquad \dfrac{\Phi, \Psi \vdash \{p\}g\{r\} \qquad \Phi \vdash r \to q \vee r}{\Phi, \Psi \vdash \{p\}g\{q \vee r\}}\,(\mathsf{weak})}{\Phi, \Psi \vdash \{p\}f \sqcap g\{q \vee r\}}\,(\mathsf{dem}).$$

Notice the similarity of the rule (dem') with the definition of the semantic demonic choice operation $\sqcap$ in Figure 6.

Observation 5.8 (Weakening The Trivial Rules)**.** In the Hoare-style calculus of Figure 7 we included two "trivial" axioms:

$$\frac{}{\Phi, \Psi \vdash \{\mathsf{false}\}f\{q\}}\,(\mathsf{join}_0) \qquad \frac{}{\Phi, \Psi \vdash \{p\}f\{\mathsf{true}\}}\,(\mathsf{meet}_0)$$

We claim that they can be weakened into the axioms

$$\frac{}{\Phi, \Psi \vdash \{\mathsf{false}\}a\{q\}}\,(a\text{-}\mathsf{join}_0) \qquad \frac{}{\Phi, \Psi \vdash \{p\}a\{\mathsf{true}\}}\,(a\text{-}\mathsf{meet}_0)$$

so that they apply only to atomic programs $a, b, \ldots$, without changing the theory generated by the calculus. The claim is that if we replace (join_0) and (meet_0) by the weaker axioms $(a\text{-}\mathsf{join}_0)$ and $(a\text{-}\mathsf{meet}_0)$, then we can still prove (join_0) and (meet_0) for arbitrary terms.

Proof. Suppose that $\vdash_w$ denotes provability in the weakened proof system with $(a\text{-}\mathsf{join}_0)$ and $(a\text{-}\mathsf{meet}_0)$. We claim that for every program term f and all tests p, q, it holds:

$$\vdash_w \{\mathsf{false}\}f\{q\} \qquad \text{and} \qquad \vdash_w \{p\}f\{\mathsf{true}\}.$$

It suffices to establish that $\vdash_w \{\mathsf{false}\}f\{\mathsf{false}\}$ and $\vdash_w \{\mathsf{true}\}f\{\mathsf{true}\}$, because we have:

$$\frac{\vdash_w \{\mathsf{false}\}f\{\mathsf{false}\} \qquad \vdash \mathsf{false} \to q}{\vdash_w \{\mathsf{false}\}f\{q\}}\,(\mathsf{weak}) \qquad \frac{\vdash_w \{\mathsf{true}\}f\{\mathsf{true}\} \qquad \vdash p \to \mathsf{true}}{\vdash_w \{p\}f\{\mathsf{true}\}}\,(\mathsf{weak})$$

The proof is by induction on the structure of f. We will only give the following derivation

$$\dfrac{\dfrac{\dfrac{\mathsf{false} \wedge p \to \mathsf{false} \qquad \{\mathsf{false}\}f\{\mathsf{false}\}\ (\text{I.H.})}{\{\mathsf{false} \wedge p\}f\{\mathsf{false}\}}\,(\mathsf{weak})}{\{\mathsf{false}\}\mathsf{W}pf\{\mathsf{false} \wedge \neg p\}}\,(\mathsf{loop}) \qquad \mathsf{false} \wedge \neg p \to \mathsf{false}}{\{\mathsf{false}\}\mathsf{W}pf\{\mathsf{false}\}}\,(\mathsf{weak})$$

as an illustrative example. The other cases equally straightforward and we omit them. □

Theorem 5.9 (Soundness)**.** *The Hoare calculus of Figure 7 is sound.*

Proof. The soundness of the proposed Hoare calculus is an immediate consequence of the following properties that are formulated at a purely semantic level.

$$\{P\}\mathbf{1}_S\{P\} \qquad \{P\}\mathbf{0}_S\{Q\} \qquad \frac{\{P\}\phi\{Q\} \qquad \{Q\}\psi\{R\}}{\{P\}\phi;\psi\{R\}} \qquad \frac{\{Q \cap P\}\phi\{R\} \qquad \{Q \cap {\sim}P\}\psi\{R\}}{\{Q\}P[\phi,\psi]\{R\}}$$

$$\frac{\{R \cap P\}\phi\{R\}}{\{R\}\mathbf{wh}\,P\,\mathbf{do}\,\phi\{R \cap {\sim}P\}} \qquad \frac{\{P\}\phi_i\{Q\}}{\{P\}\phi_1 \cup \phi_2\{Q\}} \qquad \frac{\{P\}\phi_\kappa\{Q\}}{\{P\}\bigcap_\kappa \phi_\kappa\{Q\}}$$

$$\frac{P' \subseteq P \qquad \{P\}\phi\{Q\} \qquad Q \subseteq Q'}{\{P'\}\phi\{Q'\}} \qquad \frac{\{P_1\}\phi\{Q\} \qquad \{P_2\}\phi\{Q\}}{\{P_1 \cup P_2\}\phi\{Q\}} \qquad \frac{\{\emptyset\}\phi\{Q\}}{\{P\}\phi\{S\}}$$

For predicates $P, Q \subseteq S$ and a game function $\phi : S \mathrel{\leadsto} S$, we understand $\{P\}\phi\{Q\}$ as the assertion saying that $(u, Q) \in \phi$ for every state $u \in P$. Establishing the above semantic properties of game functions is a tedious but straightforward task. We will therefore only consider here the case $\mathbf{wh}\,P\,\mathbf{do}\,\phi$ and leave the rest to the reader. Recall the definition:

$$\mathbf{wh}\,P\,\mathbf{do}\,\phi = \textstyle\bigcap_{\kappa \in \mathbf{Ord}} W_\kappa \qquad W_0 = P[\mathbf{0}_S, \mathbf{1}_S]$$
$$W_{\kappa+1} = P[\phi; W_\kappa, \mathbf{1}_S] \qquad W_\lambda = \textstyle\bigcap_{\kappa<\lambda} W_\kappa, \text{ for limit ordinal } \lambda$$

We show by transfinite induction that $\{R\}W_\kappa\{R \cap \neg P\}$. Indeed, for the base case W_0 and for the case of the successor ordinal $W_{\kappa+1}$ we have the following derivations:

$$\frac{\{R \cap P\}\mathbf{0}_S\{R \cap {\sim}P\} \qquad \{R \cap {\sim}P\}\mathbf{1}_S\{R \cap {\sim}P\}}{\{R\}P[\mathbf{0}_S, \mathbf{1}_S]\{R \cap {\sim}P\}}$$

$$\dfrac{\dfrac{\{R \cap P\}\phi\{R\}\ (\text{hyp.}) \qquad \{R\}W_\kappa\{R \cap {\sim}P\}\ (\text{I.H.})}{\{R \cap P\}\phi; W_\kappa\{R \cap {\sim}P\}} \qquad \{R \cap {\sim}P\}\mathbf{1}_S\{R \cap {\sim}P\}}{\{R\}P[\phi; W_\kappa, \mathbf{1}_S]\{R \cap {\sim}P\}}$$

The case W_λ of the limit ordinal λ is handled using the I.H. for each ordinal $\kappa < \lambda$ and the infinitary rule for $\bigcap$. Finally, the assertion $\{R\}\mathbf{wh}\,P\,\mathbf{do}\,\phi\{R \cap {\sim}P\}$ is shown using the claim and the rule for infinitary intersection. □

Example 5.10. We will use the Hoare logic of Figure 7 to establish the partial-correctness property $\{x = 0\}h\{x = 1\}$ for the program h of Example 3.8 (recall the abbreviations f, g).

1. $\{x = 0\}\mathsf{id}\{x = 0\}$	[skip]	$\{\mathsf{Precondition} : x = 0\}$
2. $\{x = 0\}\mathsf{id} \sqcup x\mathsf{++}\{x = 0\}$	[1, ang]	// invariant $inv \triangleq (x = 0) \vee (x = 1)$
3. $\{x = 0\}x\mathsf{++}\{x = 1\}$	[hypothesis]	$\mathsf{while}\ (x = 0)\ \mathsf{do}$
4. $\{x = 0\}\mathsf{id} \sqcap x\mathsf{++}\{inv\}$	[1, 3, dem']	$\quad // \ inv \wedge (x = 0) \leftrightarrow (x = 0)$
5. $\{x = 0\}f; g\{inv\}$	[2, 4, seq]	$\quad \mathsf{id} \sqcup x\mathsf{++}$
6. $\{inv \wedge (x = 0)\}f; g\{inv\}$	[5, bool, weak]	$\quad \mathsf{id} \sqcap x\mathsf{++}$
7. $\{inv\}h\{inv \wedge (x \neq 0)\}$	[6, loop]	$\quad // \ (x = 0) \vee (x = 1)$
8. $\{x = 0\}h\{x = 1\}$	[7, bool, weak]	$\{\mathsf{Postcondition} : x = 1\}$

The only hypothesis for atomic symbols used in the proof is $\{x = 0\}x\mathsf{++}\{x = 1\}$.

6. First Completeness Theorem: Weak Hoare Theory

We will now prove the completeness of the Hoare calculus of Figure 7 with respect to the class of all interpretations. This means that we consider arbitrary interpretations of the atomic programs $a, b, \ldots$ as game functions. So, the deductive system of Figure 7 is complete for the weak Hoare theory of while game schemes. Note that this is an *unconditional* completeness result (no extra assumptions about expressiveness or about the first-order theory of the domain of computation), not a relative completeness theorem in the sense of [Coo78].

We show our result by constructing a "free" interpretation $I_{\Phi\Psi}$ from the hypotheses Φ and Ψ about the atomic symbols. We can think of this interpretation as the *least restrictive* interpretation that satisfies the hypotheses. Completeness follows from the fact that the interpretation $I_{\Phi\Psi}$ characterizes the theory generated by our calculus. In other words, everything that is true in $I_{\Phi\Psi}$ is provable using our partial-correctness calculus.

Definition 6.1 (The Free Game Interpretation)**.** Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. We define the *free game interpretation* $I_{\Phi\Psi}$ (w.r.t. Φ and Ψ) to have At_Φ as state space, and to interpret the tests as I_Φ (the free test interpretation w.r.t. Φ, see Definition 5.5) does. Moreover, the interpretation $I_{\Phi\Psi}(a) : \mathsf{At}_\Phi \rightsquigarrow \mathsf{At}_\Phi$ of the atomic action a is given by: for every Φ-consistent atom α,

– $(\alpha, \mathsf{At}_\Phi) \in I_{\Phi\Psi}(a)$, and for every subset $X \subsetneq \mathsf{At}_\Phi$,
– $(\alpha, X) \in I_{\Phi\Psi}(a)$ iff there exists $\{p\}a\{q\} \in \Psi$ s.t. $\alpha \leq p$ and $I_\Phi(q) \subseteq X$.

Lemma 6.2. Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. The free game interpretation $I_{\Phi\Psi}$ satisfies all formulas in Φ and Ψ. □

Theorem 6.3 (Completeness)**.** *Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. For every program term f and every Φ-consistent atom α,*

$$(\alpha, X) \in I_{\Phi\Psi}(f) \text{ implies that } \Phi, \Psi \vdash \{\alpha\}f\{\textstyle\bigvee X\}.$$

Proof. The proof proceeds by induction on the structure of the program term f. Recall that we have assumed having a complete Boolean calculus (see Note 5.6).

We begin with the base case of the skip program id. Consider an arbitrary pair (α, X) of $I_{\Phi\Psi}(\mathsf{id})$, where $\alpha \in X$. Since $I_{\Phi\Psi}(\mathsf{id}) = \mathbf{1}_{\mathsf{At}_\Phi}$, we know that $\alpha \in X$. Using the (skip) axiom and the weakening rule, we have the derivation:

$$\dfrac{\dfrac{}{\Phi, \Psi \vdash \{\alpha\}\mathsf{id}\{\alpha\}}\,(\mathsf{skip}) \quad \dfrac{\alpha \in X \subseteq \mathsf{At}_\Phi}{\Phi \vdash \alpha \to \bigvee X}}{\Phi, \Psi \vdash \{\alpha\}\mathsf{id}\{\bigvee X\}}\,(\mathsf{weak}).$$

Now, we handle the case of the always diverging program $\bot$. Let (α, X) be an arbitrary element of $I_{\Phi\Psi}(\bot) = \mathbf{0}_{\mathsf{At}_\Phi} = \mathsf{At}_\Phi \times \wp\mathsf{At}_\Phi$. The (dvrg) axiom gives us immediately

$$\dfrac{}{\Phi, \Psi \vdash \{\alpha\}\bot\{\bigvee X\}}\,(\mathsf{dvrg}).$$

For the case of an atomic action a, consider an arbitrary pair (α, X) in $I_{\Phi\Psi}(a)$. If $X = \mathsf{At}_\Phi$, then we have the following derivation:

$$\dfrac{\dfrac{}{\Phi, \Psi \vdash \{\alpha\}a\{\mathsf{true}\}}\,(\mathsf{meet}_0) \quad \dfrac{I_\Phi(\mathsf{true}) = \mathsf{At}_\Phi}{\Phi \vdash \mathsf{true} \to \bigvee \mathsf{At}_\Phi}}{\Phi, \Psi \vdash \{\alpha\}a\{\bigvee \mathsf{At}_\Phi\}}\,(\mathsf{weak}).$$

Assume now that $X \subsetneq \mathsf{At}_\Phi$. By definition of $I_{\Phi\Psi}(a)$, there exists a simple Hoare hypothesis $\{p\}a\{q\}$ in Ψ such that $\alpha \leq p$ and $I_\Phi(q) \subseteq X$. So,

$$\dfrac{\dfrac{\alpha \leq p}{\Phi \vdash \alpha \to p} \quad \dfrac{\{p\}a\{q\} \text{ in } \Psi}{\Phi, \Psi \vdash \{p\}a\{q\}}\,(\mathsf{hyp}) \quad \dfrac{I_\Phi(q) \subseteq X \subseteq \mathsf{At}_\Phi}{\Phi \vdash q \to \bigvee X}}{\Phi, \Psi \vdash \{\alpha\}a\{\bigvee X\}}\,(\mathsf{weak}).$$

This concludes the proof for the case of atomic programs.

We will handle now the case $f; g$ of sequential composition. Let (α, Y) be an arbitrary pair in $I_{\Phi\Psi}(f; g) = I_{\Phi\Psi}(f)\mathbin{;} I_{\Phi\Psi}(g)$. By definition of the $\mathbin{;}$ operation on game functions, there exists $X \subseteq \mathsf{At}_\Phi$ such that $(\alpha, X) \in I_{\Phi\Psi}(f)$, and $(\beta, Y) \in I_{\Phi\Psi}(g)$ for every $\beta \in X$. So,

$$\dfrac{\dfrac{(\alpha, X) \text{ in } I_{\Phi\Psi}(f)}{\Phi, \Psi \vdash \{\alpha\}f\{\bigvee X\}}\,(\text{I.H.}) \quad \dfrac{\dfrac{(\beta, Y) \text{ in } I_{\Phi\Psi}(g)}{\Phi, \Psi \vdash \{\beta\}g\{\bigvee Y\}}\,(\text{I.H.}) \qquad \beta \in X}{\Phi, \Psi \vdash \{\bigvee X\}g\{\bigvee Y\}}\,(\mathsf{join})}{\Phi, \Psi \vdash \{\alpha\}f; g\{\bigvee Y\}}\,(\mathsf{seq}).$$

Observe in the derivation above that we may have to apply the (join) rule several times (finitely many), because X may contain several Φ-consistent atoms.

For the case of the conditional $\mathsf{if}\, p \,\mathsf{then}\, f \,\mathsf{else}\, g$, let us consider a pair (α, X) in $I_{\Phi\Psi}(p[f, g])$. We deal with the case where $\alpha \leq p$. We obtain the following derivations:

$$\dfrac{\Phi \vdash \alpha \wedge p \to \alpha \quad \dfrac{(\alpha, X) \text{ in } I_{\Phi\Psi}(f)}{\Phi, \Psi \vdash \{\alpha\}f\{\bigvee X\}}\,(\text{I.H.})}{(1)\ \Phi, \Psi \vdash \{\alpha \wedge p\}f\{\bigvee X\}}\,(\mathsf{weak})$$

$$\dfrac{\dfrac{\alpha \leq p}{\Phi \vdash \alpha \wedge \neg p \to \mathsf{false}} \quad \dfrac{}{\Phi, \Psi \vdash \{\mathsf{false}\}g\{\bigvee X\}}\,(\mathsf{join}_0)}{(2)\ \Phi, \Psi \vdash \{\alpha \wedge \neg p\}g\{\bigvee X\}}\,(\mathsf{weak})$$

$$\dfrac{(1)\ \dfrac{\vdots}{\Phi, \Psi \vdash \{\alpha \wedge p\}f\{\bigvee X\}} \quad (2)\ \dfrac{\vdots}{\Phi, \Psi \vdash \{\alpha \wedge \neg p\}g\{\bigvee X\}}}{\Phi, \Psi \vdash \{\alpha\}\mathsf{if}\, p \,\mathsf{then}\, f \,\mathsf{else}\, g\{\bigvee X\}}\,(\mathsf{cond})$$

The proof for the case where $\alpha \leq \neg p$ is completely analogous.

We handle now the case of the loop $\mathsf{W}pf$. Let (γ, Γ) be an arbitrary pair in the game function $I_{\Phi\Psi}(\mathsf{W}pf) = \mathbf{wh}\, I_\Phi(p)\, \mathbf{do}\, I_{\Phi\Psi}(f) = \bigcap_i W_i$, where the sequence W_i is given by

$$W_0 = I_\Phi(p)\big[\mathbf{0}_S, \mathbf{1}_S\big] \qquad\qquad W_{i+1} = I_\Phi(p)\big[I_{\Phi\Psi}(f); W_i, \mathbf{1}_S\big]$$

We do not need to consider the entire transfinite sequence $(W_\kappa)_{\kappa \in \mathbf{Ord}}$, because the space of game functions on At_Φ is finite and hence the sequence stabilizes in a finite number of steps. Define the sequence $(V_i)_{i \geq 0}$ by

$$V_i = \{\alpha \in \mathsf{At}_\Phi \mid (\alpha, \Gamma) \in W_i\}.$$

We give an inductive definition for $(V_i)_{i \geq 0}$ that is equivalent to the above:

$$\begin{aligned}
V_0 &= \{\alpha \in \mathsf{At}_\Phi \mid (\alpha, \Gamma) \in W_0\} \\
&= \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p \text{ or } (\alpha \leq \neg p \text{ and } \alpha \in \Gamma)\} \\
&= \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p \text{ or } \alpha \in \Gamma\} \\
&= I_\Phi(p) \cup (\sim I_\Phi(p) \cap \Gamma) \\
V_{i+1} &= \{\alpha \in \mathsf{At}_\Phi \mid (\alpha, \Gamma) \in W_{i+1}\} \\
&= \{\alpha \in \mathsf{At}_\Phi \mid (\alpha \leq p \text{ and } (\alpha, \Gamma) \in I_{\Phi\Psi}(f); W_i) \text{ or } (\alpha \leq \neg p \text{ and } \alpha \in \Gamma)\} \\
&= \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p \text{ and } (\alpha, \Gamma) \in I_{\Phi\Psi}(f); W_i\} \cup (\sim I_\Phi(p) \cap \Gamma) \\
&= \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p \text{ and } (\alpha, V_i) \in I_{\Phi\Psi}(f)\} \cup (\sim I_\Phi(p) \cap \Gamma)
\end{aligned}$$

The last equality above is justified by the following equivalences:

$$\begin{aligned}
&(\alpha, \Gamma) \in I_{\Phi\Psi}(f); W_i \iff \\
&\exists Y.\ (\alpha, Y) \in I_{\Phi\Psi}(f), \text{ and } (\beta, \Gamma) \in W_i \text{ for every } \beta \in Y \\
&\exists Y.\ (\alpha, Y) \in I_{\Phi\Psi}(f), \text{ and } \beta \in V_i \text{ for every } \beta \in Y \\
&\exists Y.\ (\alpha, Y) \in I_{\Phi\Psi}(f) \text{ and } Y \subseteq V_i,
\end{aligned}$$

which is equivalent to $(\alpha, V_i) \in I_{\Phi\Psi}(f)$. Define the sequence $(U_i)_{i \geq 0}$ by

$$\begin{aligned}
U_0 &= \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq \neg p \text{ and } \alpha \notin \Gamma\} \\
&= \sim I_\Phi(p) \cap \sim \Gamma \\
U_{i+1} &= U_0 \cup \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p \text{ and } \forall Y \text{ with } (\alpha, Y) \in I_{\Phi\Psi}(f)\colon\ Y \cap U_i \neq \emptyset\}
\end{aligned}$$

Intuitively, the set U_i gives us the atoms from which the demon can force the execution towards the "error states" U_0 in at most i iterations of the loop.

Claim. For every $i \geq 0$, it holds that $V_i = \sim U_i = \mathsf{At}_\Phi \setminus U_i$.

Now, we define $U = \bigcup_{i \geq 0} U_i$ and $V = \bigcap_{i \geq 0} V_i$. The above claim implies that $V = \sim U$. Moreover, since $I_{\Phi\Psi}(\mathsf{W}pf) = \bigcap_i W_i$, it is easy to see that

$$V = \{\alpha \in \mathsf{At}_\Phi \mid (\alpha, \Gamma) \in I_{\Phi\Psi}(\mathsf{W}pf)\}.$$

Our hypothesis $(\gamma, \Gamma) \in I_{\Phi\Psi}(\mathsf{W}pf)$ then gives us that $\gamma \in V$.

Claim. If $\alpha \leq p$ and $\alpha \in V$, then (α, V) is in $I_{\Phi\Psi}(f)$.

So, we have the following derivations, where the first one is for an arbitrary Φ-consistent atom $\alpha \in V \cap I_\Phi(p)$:

$$\dfrac{\dfrac{\Phi \vdash (\bigvee V) \wedge p \to \bigvee(V \cap I_\Phi(p)) \qquad \dfrac{\dfrac{(\alpha, V) \text{ in } I_{\Phi\Psi}(f)}{\Phi, \Psi \vdash \{\alpha\} f \{\bigvee V\}}\,(\text{I.H.}) \qquad \alpha \in V \cap I_\Phi(p)}{\Phi, \Psi \vdash \{\bigvee(V \cap I_\Phi(p))\} f \{\bigvee V\}}\,(\mathsf{join})}{\Phi, \Psi \vdash \{(\bigvee V) \wedge p\} f \{\bigvee V\}}}{(1)\ \Phi, \Psi \vdash \{\bigvee V\} \mathsf{while}\, p\, \mathsf{do}\, f \{(\bigvee V) \wedge \neg p\}}\,(\mathsf{loop})$$

$$\dfrac{\dfrac{\gamma \in V}{\Phi \vdash \gamma \to r} \quad (1)\ \dfrac{\vdots \quad r \triangleq \bigvee V}{\Phi, \Psi \vdash \{r\} \mathsf{W}p f \{r \wedge \neg p\}} \qquad \dfrac{V \cap {\sim} I_\Phi(p) = \Gamma \cap {\sim} I_\Phi(p)}{\Phi \vdash r \wedge \neg p \to \bigvee \Gamma}}{\Phi, \Psi \vdash \{\gamma\} \mathsf{while}\, p\, \mathsf{do}\, f \{\bigvee \Gamma\}}$$

The last deduction step is done using the weakening rule.

For angelic choice $f \sqcup g$, let (α, X) be a pair in $I_{\Phi\Psi}(f \sqcup g) = I_{\Phi\Psi}(f) \sqcup I_{\Phi\Psi}(g)$, which is equal to $I_{\Phi\Psi}(f) \cup I_{\Phi\Psi}(g)$. We assume that (α, X) is in $I_{\Phi\Psi}(f)$.

$$\dfrac{\dfrac{(\alpha, X) \text{ in } I_{\Phi\Psi}(f)}{\Phi, \Psi \vdash \{\alpha\} f \{\bigvee X\}}\,(\text{I.H.})}{\Phi, \Psi \vdash \{\alpha\} f \sqcup g \{\bigvee X\}}\,(\mathsf{ang}_1).$$

The case of $(\alpha, X) \in I_{\Phi\Psi}(g)$ is handled analogously.

For demonic choice $f \sqcap g$, let $(\alpha, X \cup Y)$ be a pair in $I_{\Phi\Psi}(f \sqcap g) = I_{\Phi\Psi}(f) \sqcap I_{\Phi\Psi}(g)$, where $(\alpha, X) \in I_{\Phi\Psi}(f)$ and $(\alpha, Y) \in I_{\Phi\Psi}(g)$. We obtain the derivation:

$$\dfrac{\dfrac{(\alpha, X) \text{ in } I_{\Phi\Psi}(f)}{\Phi, \Psi \vdash \{\alpha\} f \{\bigvee X\}}\,(\text{I.H.}) \qquad \dfrac{X \subseteq X \cup Y \subseteq \mathsf{At}_\Phi}{\Phi \vdash \bigvee X \to \bigvee(X \cup Y)}}{\Phi, \Psi \vdash \{\alpha\} f \{\bigvee(X \cup Y)\}}\,(\mathsf{weak})$$

and similarly we also get that $\Phi, \Psi \vdash \{\alpha\} g \{\bigvee(X \cup Y)\}$. Finally,

$$\dfrac{\Phi, \Psi \vdash \{\alpha\} f \{\bigvee(X \cup Y)\} \qquad \Phi, \Psi \vdash \{\alpha\} g \{\bigvee(X \cup Y)\}}{\Phi, \Psi \vdash \{\alpha\} f \sqcap g \{\bigvee(X \cup Y)\}}\,(\mathsf{dem})$$

by the rule for demonic choice, and we are done. □

Corollary 6.4 (Completeness)**.** *Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. For every program f, the following are equivalent:*

(1) $\Phi, \Psi \models \{p\} f \{q\}$.
(2) *For every Φ-consistent $\alpha \leq p$, the pair $(\alpha, I_\Phi(q))$ is in $I_{\Phi\Psi}(f)$.*
(3) $\Phi, \Psi \vdash \{p\} f \{q\}$.

Proof. For the implication (1) ⇒ (2), recall that the free interpretation $I_{\Phi\Psi}$ satisfies the hypotheses in Φ and Ψ (Lemma 6.2). So, it must be that $I_{\Phi\Psi}$ satisfies $\{p\} f \{q\}$. For a Φ-consistent atom with $\alpha \leq p$, we have that $I_{\Phi\Psi}, \alpha \models p$ and hence $(\alpha, I_{\Phi\Psi}(q))$ is in $I_{\Phi\Psi}(f)$. But $I_{\Phi\Psi}(q) = I_\Phi(q)$, and we thus conclude that $(\alpha, I_\Phi(q)) \in I_{\Phi\Psi}(f)$.

We will prove now the implication (2) $\Rightarrow$ (3). Theorem 6.3 says: $(\alpha, I_\Phi(q)) \in I_{\Phi\Psi}(f)$ implies that $\Phi, \Psi \vdash \{\alpha\} f \{\bigvee I_\Phi(q)\}$. So, we have the following deduction

$$\dfrac{\dfrac{\dfrac{(\alpha, I_\Phi(q)) \text{ in } I_{\Phi\Psi}(f)}{\Phi, \Psi \vdash \{\alpha\} f \{\bigvee I_\Phi(q)\}}\,(\text{Thm } 6.3) \qquad \Phi \vdash (\bigvee I_\Phi(q)) \to q}{\Phi, \Psi \vdash \{\alpha\} f \{q\}} \quad \begin{array}{l}\text{for } \alpha \in \mathsf{At}_\Phi \\ \text{with } \alpha \leq p\end{array}}{\Phi, \Psi \vdash \{\bigvee I_\Phi(p)\} f \{q\}}\,(\mathsf{join}),$$

because $I_\Phi(p) = \{\alpha \in \mathsf{At}_\Phi \mid \alpha \leq p\}$. Finally, notice that $\Phi \vdash p \to \bigvee I_\Phi(p)$ and by the weakening rule we conclude that $\Phi, \Psi \vdash \{p\} f \{q\}$.

The implication (3) $\Rightarrow$ (1) is the soundness result for our Hoare calculus, which we have already proved in Theorem 5.9. □

Corollary 6.4 gives us a decision procedure for the weak Hoare theory of dual nondeterminism. Given a Hoare implication $\Phi, \Psi \Rightarrow \{p\} f \{q\}$, we simply have to compute the free interpretation $I_{\Phi\Psi}(f) \subseteq \mathsf{At}_\Phi \times \wp\mathsf{At}_\Phi$, which is a finite object. Observe that $I_{\Phi\Psi}(f)$ is of doubly exponential size. We will see later that, with some more work, we can devise a faster algorithm of exponential complexity.

7. Strong Hoare Theory: Completeness and Complexity

The completeness theorem of §6 concerns the theory generated by the class of all interpretations, that is, when the atomic programs are allowed to be interpreted as any game function. However, for most realistic applications the atomic actions $a, b, \ldots$ correspond to computational operations (e.g., variable assignments $x := t$, etc.) that involve no angelic nondeterministic choice. This leads us to consider a strictly smaller class of interpretations, and the question is raised of whether this smaller class has the same Hoare theory. This section is devoted to the in-depth study of the theory over this subclass of interpretations. We obtain both an unconditional completeness theorem and a complexity characterization.

Definition 7.1 (Validity Over a Class of Interpretations)**.** We fix a language with atomic tests and atomic actions. Let $\mathcal{C}$ be a class of interpretations of the atomic symbols (extending to all tests and programs in the usual way). We say that a simple Hoare implication $\Phi, \Psi \Rightarrow \{p\} f \{q\}$ is *valid in* $\mathcal{C}$ (or $\mathcal{C}$*-valid*) if every interpretation I in $\mathcal{C}$ satisfies the implication. We then write $\Phi, \Psi \models_{\mathcal{C}} \{p\} f \{q\}$. The set of all $\mathcal{C}$-validities is called the *Hoare theory of* $\mathcal{C}$.

Let *All* be the class of all interpretations. Observe that an implication is valid iff it is valid in *All*. Now, let *Dem* $\subseteq$ *All* be the strict subclass of interpretations where the atomic actions are interpreted as non-angelic game functions.

Lemma 7.2 (Soundness)**.** The rule (meet) of Figure 8, where a is an atomic action, is sound for the class *Dem* of interpretations.

Proof. Let I be an interpretation in the class *Dem*, which means that the game function $I(a) : S \rightsquigarrow S$ is non-angelic. Suppose that I satisfies the premises of the rule (meet), and also that it satisfies the hypotheses Φ and Ψ. It follows that I satisfies the assertions $\{p\} a \{q_1\}$ and $\{p\} a \{q_2\}$. We have to show that I also satisfies the assertion $\{p\} a \{q_1 \wedge q_2\}$. Let u be a state with $u \in I(p)$. Then, we have that $(u, I(q_1)) \in I(a)$ and $(u, I(q_2)) \in I(a)$. Since $I(a)$ is non-angelic, there exists a unique subset $X \subseteq S$ such that $I(a)(u) = \{Y \subseteq S \mid X \subseteq Y\}$. But $I(q_1)$ and $I(q_2)$ are both in $I(a)(u)$, which means that $X \subseteq I(q_1)$ and $X \subseteq I(q_2)$. We

$$\frac{\Phi, \Psi \vdash \{p\}a\{q_1\} \qquad \Phi, \Psi \vdash \{p\}a\{q_2\}}{\Phi, \Psi \vdash \{p\}a\{q_1 \wedge q_2\}}\ (a\text{-}\mathsf{meet})$$

Figure 8: A rule that is sound when the atomic actions are interpretated as non-angelic game functions. That is, (meet) is sound for the class *Dem*.

thus obtain that $X \subseteq I(q_1) \cap I(q_2) = I(q_1 \wedge q_2)$, and therefore $(u, I(q_1 \wedge q_2)) \in I(a)$. So, $I \models \{p\}a\{q_1 \wedge q_2\}$, and the proof is complete. □

Lemma 7.2 also establishes that the Hoare theory of *Dem* is different from the Hoare theory of *All*. Strictly more implications hold, when we restrict attention to the interpretations of *Dem*. For example, consider the set of hypotheses Ψ, which consists of the two simple assertions $\{p\}a\{q\}$ and $\{p\}a\{r\}$, where p, q, r are distinct atomic tests. Observe that the implication $\Psi \Rightarrow \{p\}a\{q \wedge r\}$ is valid in *Dem* (by Lemma 7.2), but it is not valid in *All* (by virtue of Corollary 6.4).

Definition 7.3 (The Free Non-Angelic Interpretation)**.** Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. For an atomic action a, define the nondeterministic interpretation $R_{\Phi\Psi}(a) : \mathsf{At}_\Phi \to \wp\mathsf{At}_\Phi$ as

$$R_{\Phi\Psi}(a)(\alpha) \triangleq \{\beta \in \mathsf{At}_\Phi \mid \text{for every } \{p\}a\{q\} \in \Psi,\ \alpha \leq p \text{ implies that } \beta \leq q\}.$$

We define the *free non-angelic interpretation* $J_{\Phi\Psi}$ (w.r.t. Φ and Ψ) to have At_Φ as state space, and to interpret the tests as I_Φ (the free test interpretation w.r.t. Φ, see Definition 5.5) does. Moreover, the interpretation $J_{\Phi\Psi}(a) : \mathsf{At}_\Phi \rightsquigarrow \mathsf{At}_\Phi$ of the atomic action a is the lifting of $R_{\Phi\Psi}(a)$, that is, it is given by $J_{\Phi\Psi}(a) = \mathsf{lift}\, R_{\Phi\Psi}(a)$.

Lemma 7.4. Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. The free non-angelic interpretation $J_{\Phi\Psi}$ satisfies both Φ and Ψ. □

Recall that we used the symbol $\vdash$ in §5 to denote provability in the Hoare-style system of Figure 7. Now, we will use the symbol $\vdash_d$ to denote provability in the system that extends the calculus of Figure 7 with the additional rule (meet) shown in Figure 8.

Theorem 7.5 (Completeness)**.** *Let Φ be a finite set of tests, and Ψ be a finite set of simple Hoare assertions. For every program term f and every Φ-consistent atom α,*

$$(\alpha, Y) \in J_{\Phi\Psi}(f) \text{ implies that } \Phi, \Psi \vdash_d \{\alpha\}f\{\textstyle\bigvee Y\}.$$

Proof. We will only consider the base case of an atomic program a. All the other cases are handled exactly as in Theorem 6.3, so we omit them. Let α be a Φ-consistent atom. Define

$$\begin{aligned} X = R_{\Phi\Psi}(a)(\alpha) &= \{\beta \in \mathsf{At}_\Phi \mid \text{for all } \{p\}a\{q\} \in \Psi\colon\ \alpha \leq p \text{ implies } \beta \leq q\} \\ &= I_\Phi\big(\textstyle\bigwedge Q\big), \text{ where } Q = \{q \mid \{p\}a\{q\} \in \Psi \text{ and } \alpha \leq p\}. \end{aligned}$$

The claim is that $\Phi, \Psi \vdash_d \{\alpha\}a\{\bigvee X\}$. If the set Q of tests (defined above) is empty, then $\bigwedge Q = \mathsf{true}$ and $X = I_\Phi(\bigwedge Q) = \mathsf{At}_\Phi$. We have the derivation

$$\frac{\dfrac{}{\Phi, \Psi \vdash_d \{\alpha\}a\{\mathsf{true}\}}\ (\mathsf{meet}_0) \qquad \Phi \vdash \mathsf{true} \to \textstyle\bigvee \mathsf{At}_\Phi}{\Phi, \Psi \vdash_d \{\alpha\}a\{\mathsf{true}\}}\ (\mathsf{weak}).$$

Now, we can assume that Q is not empty. Using the extra rule (meet) we obtain

$$\dfrac{\dfrac{\alpha \leq p}{\Phi \vdash \alpha \to p} \quad \dfrac{\{p\}a\{q\} \text{ in } \Psi}{\Phi, \Psi \vdash_d \{p\}a\{q\}}\,(\mathsf{hyp})}{\Phi, \Psi \vdash_d \{\alpha\}a\{q\}}\,(\mathsf{weak}) \qquad \begin{array}{l}\text{for every assertion}\\ \{p\}a\{q\} \text{ in } \Psi \text{ with}\\ \alpha \leq p\end{array}}{(1)\ \Phi, \Psi \vdash_d \{\alpha\}a\{\bigwedge Q\}}\,(\mathsf{meet}).$$

Finally, from $X = I_\Phi(\bigwedge Q)$ we obtain that $\Phi \vdash \bigwedge Q \to \bigvee X$, and using the weakening rule we conclude that $\Phi, \Psi \vdash_d \{\alpha\}a\{\bigvee X\}$.

Now, let (α, Y) be an arbitrary pair in $J_{\Phi\Psi}$. It follows that $X \subseteq Y$, where X was defined in the previous paragraph. So,

$$\dfrac{\Phi, \Psi \vdash_d \{\alpha\}a\{\bigvee X\} \qquad \dfrac{X \subseteq Y \subseteq \mathsf{At}_\Phi}{\Phi \vdash \bigvee X \to \bigvee Y}\,(\mathsf{weak})}{\Phi, \Psi \vdash_d \{\alpha\}a\{\bigvee Y\}}\,(\mathsf{weak})$$

and the proof is complete. □

Corollary 7.6 (Completeness)**.** *Let Φ and Ψ be finite sets of tests and simple Hoare assertions respectively. For every program f, the following are equivalent:*

(1) $\Phi, \Psi \models_{Dem} \{p\}f\{q\}$.
(2) *For every Φ-consistent $\alpha \leq p$, the pair $(\alpha, I_\Phi(q))$ is in $J_{\Phi\Psi}(f)$.*
(3) $\Phi, \Psi \vdash_d \{p\}f\{q\}$.

Proof. Similar to the proof of Corollary 6.4. □

The results so far imply that the Hoare theory of the class *Dem*, which we also call the *strong Hoare theory* of while game schemes, can be reduced to the weak Hoare theory of the class *All* (with an exponential blowup in the size of the instance). Let $\Phi, \Psi \Rightarrow \{p\}f\{q\}$ be an arbitrary Hoare implication. W.l.o.g. the axioms in Ψ are of the form $\{\alpha\}a\{q\}$, where α is an atom and a is an atomic action. Now, define Ψ' to be the set of hypotheses that results from Ψ by replacing the axioms $\{\alpha\}a\{q_i\}$ involving α, a by a single axiom $\{\alpha\}a\{\bigwedge_i q_i\}$. The crucial observation is that the interpretation $J_{\Phi\Psi}$ is the same as $I_{\Phi\Psi'}$. Using our two completeness results of Corollary 6.4 and Corollary 7.6, it follows that $\Phi, \Psi \vdash_d \{p\}f\{q\}$ iff $\Phi, \Psi' \vdash \{p\}f\{q\}$.

Now, we will investigate the computational complexity of the strong Hoare theory of while game schemes. We prove that this theory is complete for exponential time. In order to obtain the EXPTIME upper bound, we consider an operational model that corresponds to the free game interpretation. The operational model is a safety game on a finite graph, and we can decide validity by computing the winning regions of the players. The full abstraction result of §4 says that our denotational semantics coincides in a precise sense to the operational semantics. The lower bound of EXPTIME-hardness is obtained with a reduction from alternating Turing machines with polynomially bounded tapes.

Theorem 7.7 (Complexity Upper & Lower Bound)**.** *The strong Hoare theory of while game schemes (the validities over the class Dem) is* EXPTIME*-complete.*

Proof. We first deal with the upper bound. Let Φ be a finite set of tests, Ψ be a finite set of simple Hoare assertions, and $\{p\}f\{q\}$ be a Hoare assertion. We want to decide whether the simple Hoare implication $\Phi, \Psi \Rightarrow \{p\}f\{q\}$ is valid, equivalently, whether $\Phi, \Psi \vdash \{p\}f\{q\}$. Let $X = I_\Phi(q)$. According to the completeness result of Corollary 6.4, we need to check whether $(\alpha, X) \in J_{\Phi\Psi}(f)$ for every Φ-consistent $\alpha \leq p$. By Theorem 4.11, this is

$$
\begin{array}{l}
\textbf{program} \triangleq \mathsf{while}\,(\neg halt)\,\mathsf{do} \\
\qquad\qquad \mathsf{if}\ (S_{q_1} \wedge P_{a_1})\ \mathsf{then} \\
\qquad\qquad\quad \text{take transitions from } (q_1, a_1) \\
\qquad\qquad \mathsf{else\ if}\ (S_{q_2} \wedge P_{a_2})\ \mathsf{then} \\
\qquad\qquad\quad \text{take transitions from } (q_2, a_2) \\
\qquad\qquad\ \vdots \\
\qquad\qquad \mathsf{else\ if}\ (S_{q_m} \wedge P_{a_m})\ \mathsf{then} \\
\qquad\qquad\quad \text{take transitions from } (q_m, a_m) \\
\qquad\qquad \mathsf{else\ id}
\end{array}
$$

Figure 9: While game scheme that encodes the behavior of an alternating Turing machine.

equivalent to $I_\Phi(p) \times \{f\}$ being contained in the winning region of Player $\exists$ in the safety game $G_{J_{\Phi\Psi}}(f, {\sim}X)$. Observe in the proof of Theorem 4.11 that the full abstraction result remains unchanged if in the safety game $G_{J_{\Phi\Psi}}(f, {\sim}X)$ we only consider the vertices

$$
\begin{aligned}
V &= (\mathsf{At}_\Phi \times C(f)) \cup (\mathcal{X} \times C(f)), \text{ where} \\
\mathcal{X} &= \{R_{\Phi\Psi}(a)(\alpha) \mid \text{atomic action } a \in C(f) \text{ and } \alpha \in \mathsf{At}_\Phi\}.
\end{aligned}
$$

With this modification, the game $G_{J_{\Phi\Psi}}(f, {\sim}X)$ is of size exponential in the size of the input: there are exponentially many Φ-consistent atoms, and linearly many terms in $C(f)$ (see Part (1) of Lemma 3.6). We can compute the winning regions of $G_{J_{\Phi\Psi}}(f, {\sim}X)$ in time polynomial in the size of the game. So, overall we need time exponential in the size of the input to decide whether the implication is valid.

We can prove the lower bound by encoding the computations of polynomial-space bounded alternating Turing machines [CKS81], since $\mathsf{EXPTIME} = \mathsf{APSPACE}$. An alternating machine consists of the following components: states $Q = Q_{\text{and}} \cup Q_{\text{or}}$ (partitioned into and-states & or-states), input alphabet Σ, tape alphabet Γ, blank symbol $\textvisiblespace \in \Gamma$, start state q_0, and transition relation

$$\Delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{-1, 0, +1\}).$$

We use letters $q, q', \ldots$ to range over states, and $a, b, \ldots$ to range over alphabet symbols. A transition $\langle (q, a), (q', b, d) \rangle \in \Delta$ says that if the machine is in state q and is scanning the symbol a, then it spawns a new process with its own copy of the tape in which the state is set to q', the symbol b is written over the current position, and the cursor moves by d. If $d = -1$ ($d = +1$) the cursor moves one position to the left (right), and if $d = 0$ the cursor stays in the same position. The machine accepts (rejects) if it halts at an and-state (or-state).

The idea is to simulate the alternating machine with a while program scheme that consists of a single while loop. The loop corresponds to the execution loop of the machine, and the body of the loop encodes the transition and process spawning rules (see Figure 9). Without loss of generality we can assume that every computation path halts.

We introduce atomic tests P_i^a for every tape symbol $a \in \Gamma$ and every position i. Intuitively, P_i^a is true when the tape has symbol a at position i. The hypotheses

$$\textstyle\bigwedge_i \bigvee_a P_i^a \quad \text{and} \quad \bigwedge_i \bigwedge_{a \neq b} \neg(P_i^a \wedge P_j^b)$$

say that every position is associated with a unique symbol. We also have atomic tests C_i for every position i. The test C_i is true when the cursor is scanning the i-th position of the

tape. We require that

$$\bigvee_i C_i \quad \text{and} \quad \bigwedge_{i \neq j} \neg(C_i \wedge C_j).$$

For every state $q \in Q$ of the machine, we introduce an atomic test S_q. The test S_q is true when the machine is in state q, so we demand:

$$\bigvee_q S_q \quad \text{and} \quad \bigwedge_{q \neq q'} \neg(S_q \wedge S_{q'}).$$

The machine halts when it is in a state q and the cursor is scanning a symbol a so that the pair (q, a) has no Δ-successor. In this case, we say that the pair (q, a) is a dead-end. So, we define the abbreviations

$$P_a \triangleq \bigvee_i (C_i \wedge P_i^a) \qquad\qquad \mathit{halt} \triangleq \bigvee_{q,\, a \text{ where } (q,a) \text{ is dead-end}} (S_q \wedge P_a)$$

where P_a says that the currently scanned symbol is a, and *halt* asserts that the machine can take no transition. Moreover, we define the abbreviations

$$\mathit{accept} \triangleq \mathit{halt} \wedge \Big(\bigvee_{q \in Q_{\text{and}}} S_q\Big) \qquad\qquad \mathit{reject} \triangleq \mathit{halt} \wedge \Big(\bigvee_{q \in Q_{\text{or}}} S_q\Big)$$

that describe acceptance and rejection respectively in terms of the atomic tests.

The atomic program $\mathsf{write}\ a$ writes the symbol a on the tape at the position where the cursor is, and leaves everything else unchanged. So, we take the following hypotheses for it:

$$\{C_i\}\mathsf{write}\ a\{P_i^a\} \qquad\qquad \{C_i\}\mathsf{write}\ a\{C_i\}$$

$$\{C_i \wedge P_j^b\}\mathsf{write}\ a\{P_j^b\},\ \text{for } j \neq i \qquad\qquad \{S_q\}\mathsf{write}\ a\{S_q\}$$

where i, j range over all positions, b ranges over all tape symbols, and q ranges over all machine states. The atomic program $\mathsf{move}\ d$, where $d \in \{-1, 0, 1\}$, moves the cursor by d. The tape and the machine state remain unchanged.

$$\{C_i\}\mathsf{move}\ d\{C_{i+d}\} \qquad\qquad \{P_j^a\}\mathsf{move}\ d\{P_j^a\} \qquad\qquad \{S_q\}\mathsf{move}\ d\{S_q\}$$

where i ranges over all positions for which $i + d$ is also a position, j ranges over all positions, a ranges over all tape symbols, and q ranges over all machine states. Finally, we introduce the atomic program $\mathsf{switch}\ q$, which changes the state of the machine into q. The tape and the cursor position remain unchanged.

$$\{\mathsf{true}\}\mathsf{switch}\ q\{S_q\} \qquad\qquad \{C_i\}\mathsf{switch}\ q\{C_i\} \qquad\qquad \{P_i^a\}\mathsf{switch}\ q\{P_i^a\}$$

where i ranges over all positions, and a ranges over all tape symbols. Suppose that (q, a) is a state-symbol pair that has at least one Δ-successor. If it has exactly one Δ-successor (q', b, d), then we define

$$\text{take transitions from } (q, a) \triangleq \mathsf{write}\ b; \mathsf{move}\ d; \mathsf{switch}\ q'.$$

If (q, a) has exactly two Δ-successors (q_1, b_1, d_1) and (q_2, b_2, d_2), and q is an and-state, then we define

$$\begin{aligned}\text{take transitions from } (q, a) \triangleq\ & (\mathsf{write}\ b_1; \mathsf{move}\ d_1; \mathsf{switch}\ q_1) \sqcap \\ & (\mathsf{write}\ b_2; \mathsf{move}\ d_2; \mathsf{switch}\ q_2).\end{aligned}$$

In the case where (q, a) the above Δ-successors but is an or-state, we replace $\sqcap$ by $\sqcup$ in the definition. The generalization to more than two Δ-successors is straightforward.

Now, we define the term **program** in Figure 9 that encodes the execution of the alternating Turing machine. The pairs $(q_1, a_1), \ldots, (q_m, a_m)$ range over the pairs (q, a) that have

at least one Δ-successor. For an input string $x_1x_2\cdots x_n$, we define the test *start*, which encodes the initial configuration, as

$$start = S_{q_0} \wedge C_1 \wedge \big(P_1^{x_1} \wedge \cdots \wedge P_n^{x_n} \wedge P_{n+1}^{\sqcup} \wedge \cdots \wedge P_{\pi(n)}^{\sqcup}\big),$$

where q_0 is the start state, 1 is the start position, and $\pi(n)$ is the polynomial that gives the space bound of the machine. Since the space is bounded by a polynomial $\pi(n)$, there are polynomially many positions i. So, the size of the program is polynomial in the size of the machine. Finally, the claim is that the machine accepts iff

$$\Phi, \Psi \models_{Dem} \{start\}\mathbf{program}\{accept\},$$

where Φ, Ψ are the collections of our assumptions for the atomic tests and the atomic programs respectively. □

It is an immediate corollary of the above theorem that the weak Hoare theory (over the class *All*) can also be decided in exponential time.

8. A Complete Hoare-style Calculus for Synthesis

We introduce in Figure 10 a Hoare-style calculus which can be used for the deductive synthesis of $\sqcup$-free programs that satisfy a Hoare specification. It is based on the complete calculus for the Hoare theory of the class *Dem*, which contains interpretations assigning non-angelic game functions (Definition 4.1) to the atomic programs. This is the calculus of Figure 7 with the extra rule (a-meet) of Figure 8. The main differences are:

(i) The rules (join_0) and (meet_0) of Figure 7 have been weakened into the rules (a-join_0) and (a-meet_0). This is inconsequential, as we have discussed in Observation 5.8.

(i) Every conclusion $\{p\}f\{q\}$ is decorated with a $\sqcup$-free program term t, which satisfies the specification $\{p\}t\{q\}$ and implements a winning strategy for the angel in the safety game described by the assertion $\{p\}f\{q\}$.

Another difference that deserves mention is the introduction in Figure 10 of two new variants (join') and (join'') of the standard rule (join). These rules are not necessary for completeness and they can be omitted without breaking our theorems, but they are useful from a practical viewpoint. The new rules (join') and (join'') are sound, and they allow useful shortcuts in the deductive synthesis of $\sqcup$-free programs.

Theorem 8.1 (Soundness)**.** *Suppose that a judgment $\Phi, \Psi \vdash t : \{p\}f\{q\}$ is derivable using the Hoare-style calculus of Figure 10. The following hold:*

(1) *Every game interpretation I in Dem satisfies the formula $\Phi, \Psi \Rightarrow \{p\}f\{q\}$.*
(2) *Every nondeterministic interpretation R satisfies $\Phi, \Psi \Rightarrow \{p\}t\{q\}$.*
(3) *Let R be a nondeterministic interpretation, and I be the game interpretation that lifts R (see Definition 4.8). Then, $\mathsf{lift}\, R(t) \subseteq I(f)$.*

Part (3) says that $R(t)$ implements $I(f)$, which is denoted $R(t) \sqsubseteq I(f)$, when I lifts R.

Proof. Part (1) follows from the soundness of the Hoare calculus of Figure 7 (Theorem 5.9) and from Lemma 7.2 (soundness of the (a-meet) rule for interpretations in *Dem*). Part (2) asserts the soundness of a Hoare calculus for nondeterministic while schemes, whose proof can be found in [Mam14]. For Part (3), the hypothesis says that $I(a) = \mathsf{lift}\, R(a)$ for every atomic program a, and $I(p) = R(p)$ for every test (see Definition 4.8). We consider the

$$\frac{\{p\}a\{q\} \text{ in } \Psi}{\Phi, \Psi \vdash a : \{p\}a\{q\}}\,(\mathsf{hyp}) \qquad \frac{}{\Phi, \Psi \vdash \mathsf{id} : \{p\}\mathsf{id}\{p\}}\,(\mathsf{skip}) \qquad \frac{}{\Phi, \Psi \vdash \bot : \{p\}\bot\{q\}}\,(\mathsf{dvrg})$$

$$\frac{\Phi, \Psi \vdash s : \{p\}f\{q\} \quad \Phi, \Psi \vdash t : \{q\}g\{r\}}{\Phi, \Psi \vdash s; t : \{p\}f; g\{r\}}\,(\mathsf{seq}) \qquad \frac{\Phi, \Psi \vdash s : \{q \wedge p\}f\{r\} \quad \Phi, \Psi \vdash t : \{q \wedge \neg p\}g\{r\}}{\Phi, \Psi \vdash p[s, t] : \{q\}\mathsf{if}\, p\, \mathsf{then}\, f\, \mathsf{else}\, g\{r\}}\,(\mathsf{cond})$$

$$\frac{\Phi, \Psi \vdash t : \{r \wedge p\}f\{r\}}{\Phi, \Psi \vdash \mathsf{W}pt : \{r\}\mathsf{while}\, p\, \mathsf{do}\, f\{r \wedge \neg p\}}\,(\mathsf{loop})$$

$$\frac{\Phi, \Psi \vdash t : \{p\}f_i\{q\}}{\Phi, \Psi \vdash t : \{p\}f_1 \sqcup f_2\{q\}}\,(\mathsf{ang}_i) \qquad \frac{\Phi, \Psi \vdash s : \{p\}f\{q\} \qquad \Phi, \Psi \vdash t : \{p\}g\{q\}}{\Phi, \Psi \vdash s \sqcap t : \{p\}f \sqcap g\{q\}}\,(\mathsf{dem})$$

$$\frac{\Phi \vdash p' \to p \qquad \Phi, \Psi \vdash t : \{p\}f\{q\} \qquad \Phi \vdash q \to q'}{\Phi, \Psi \vdash t : \{p'\}f\{q'\}}\,(\mathsf{weak})$$

$$\frac{\Phi, \Psi \vdash t_1 : \{p_1\}f\{q\} \qquad \Phi, \Psi \vdash t_2 : \{p_2\}f\{q\}}{\Phi, \Psi \vdash p_1[t_1, t_2] : \{p_1 \vee p_2\}f\{q\}}\,(\mathsf{join}) \qquad \frac{}{\Phi, \Psi \vdash a : \{\mathsf{false}\}a\{q\}}\,(a\text{-}\mathsf{join}_0)$$

$$\frac{\Phi, \Psi \vdash a : \{p\}a\{q_1\} \qquad \Phi, \Psi \vdash a : \{p\}a\{q_2\}}{\Phi, \Psi \vdash a : \{p\}a\{q_1 \wedge q_2\}}\,(a\text{-}\mathsf{meet}) \qquad \frac{}{\Phi, \Psi \vdash a : \{p\}a\{\mathsf{true}\}}\,(a\text{-}\mathsf{meet}_0)$$

$$\frac{\Phi, \Psi \vdash t : \{p_1\}f\{q\} \quad \Phi, \Psi \vdash t : \{p_2\}f\{q\}}{\Phi, \Psi \vdash t : \{p_1 \vee p_2\}f\{q\}}\,(\mathsf{join}') \qquad \frac{\Phi, \Psi \vdash t_1 : \{p \wedge r\}f\{q\} \quad \Phi, \Psi \vdash t_2 : \{p \wedge \neg r\}f\{q\}}{\Phi, \Psi \vdash r[t_1, t_2] : \{p\}f\{q\}}\,(\mathsf{join}'')$$

Figure 10: A sound and complete Hoare-style calculus for the synthesis of programs.

"projection" of the calculus of Figure 10 to judgments of the form $t : f$, because the rest of the information is irrelevant.

$$a : a \qquad \mathsf{id} : \mathsf{id} \qquad \bot : \bot \qquad \frac{s : f \qquad t : g}{s; t : f; g} \qquad \frac{s : f \qquad t : g}{p[s, t] : p[f, g]}$$

$$\frac{t : f}{\mathsf{W}pt : \mathsf{W}pf} \qquad \frac{t : f}{t : f \sqcup g} \qquad \frac{t : g}{t : f \sqcup g} \qquad \frac{s : f \qquad t : g}{s \sqcap t : f \sqcap g} \qquad \frac{s : f \qquad t : f}{p[s, t] : f}$$

The claim is that for every derivable judgment $t : f$, we have $R(t) \sqsubseteq I(f)$, that is, $R(\phi)$ implements $I(f)$ (see Definition 4.6). Recall that $R(t) \sqsubseteq I(f)$ iff $\mathsf{lift}\, R(t) \subseteq I(a)$. The proof proceeds by induction on the derivation of $t : f$. It is a straightforward verification, where we make repeated use of Lemma 4.7. □

Theorem 8.2 (Completeness)**.** *Let Φ and Ψ be finite sets of tests and simple Hoare assertions respectively, and f be a program s.t. $\Phi, \Psi \models_{Dem} \{p\}f\{q\}$. Then, there exists a $\sqcup$-free program t such that $\Phi, \Psi \vdash t : \{p\}f\{q\}$.*

Proof. From Corollary 7.6 we get that $\Phi, \Psi \vdash_d \{p\}f\{q\}$. From Observation 5.8 we know that the rules (join_0) and (meet_0) can be weakened to $(a\text{-}\mathsf{join}_0)$ and $(a\text{-}\mathsf{meet}_0)$ without affecting the provability of the implication $\Phi, \Psi \Rightarrow \{p\}f\{q\}$. We annotate the proof according to the rules of Figure 10, and we conclude that $\Phi, \Psi \vdash t : \{p\}f\{q\}$ for some $\sqcup$-free program t. □

Finally, we will see that solving safety games on finite graphs can be reduced to deciding the *Dem*-validity of a Hoare implication involving a while game scheme that simulates the safety game. Let $G = (V, V_\exists, V_\forall, \to, E)$ be a safety game. For every vertex $u \in V$, introduce an atomic test p_u, which asserts that the token is currently on the vertex u. We take Φ to contain the following hypotheses for the atomic tests:

$$\bigvee_{u \in V} p_u \qquad \text{and} \qquad \neg(p_u \wedge p_v) \text{ for all } u, v \in V \text{ with } u \neq v.$$

The axioms of Φ say that the token is on exactly one vertex. So, we can identify the set At_Φ of Φ-consistent atoms with the set $\{p_u \mid u \in V\}$. For every vertex $u \in V$, we introduce an atomic action $u!$, which moves the token to the vertex u. So, take Ψ to contain the axioms

$$\{\mathsf{true}\}u!\{p_u\} \text{ for every } u \in V.$$

To emphasize that Φ and Ψ depend on G, let us denote them by Φ_G and Ψ_G respectively. For an arbitrary vertex $u \in V$, we define the program term

$$(\text{take transition from } u) \triangleq \begin{cases} \bigsqcup_{v \text{ with } u \to v} v!, & \text{if } u \in V_\exists \\ \bigsqcap_{v \text{ with } u \to v} v!, & \text{if } u \in V_\forall \\ v! \ (v \text{ unique successor of } u), & \text{otherwise} \end{cases}$$

Now, we define the while game scheme that describes how the safety game is played:

$$\begin{aligned} f_G = \mathsf{while}\,(\textstyle\bigvee\{p_u \mid u \in V \setminus E\})\,\mathsf{do}& \\ &\mathsf{if}\ p_u\ \mathsf{then}\ (\text{take transition from } u) \\ &\mathsf{else\ if}\ p_v\ \mathsf{then}\ (\text{take transition from } v) \\ &\quad \cdots \\ &\mathsf{else\ if}\ p_w\ \mathsf{then}\ (\text{take transition from } w) \end{aligned}$$

where $u, v, \ldots, w$ is an enumeration of the non-error vertices. Notice that our encoding implies that a play stops as soon as an error vertex is encountered.

Theorem 8.3 (Safety Games)**.** *Let $G = (V, V_\exists, V_\forall, \to, E)$ be a finite safety game. The angel has a winning strategy from $u \in V$ iff $\Phi_G, \Psi_G \vdash \{p_u\} f_G \{\mathsf{false}\}$.*

Proof. The idea is that Player $\exists$ has a winning strategy from u iff the loop never terminates. The theorem follows immediately from the completeness result of Corollary 6.4 and the operational/denotational correspondence shown in Theorem 4.11. □

9. Example: temperature controller

We will use our language of while game schemes to encode a toy example of implementing a temperature controller. The idea is that the controller (the angel) can set the heating/cooling system into one of three modes: heat, cool or off. We model this situation with the following program term:

$$\text{angel} \triangleq (m := \mathsf{heat}) \sqcup (m := \mathsf{cool}) \sqcup (m := \mathsf{off}),$$

where the variable m stores the current mode. The demon, on the other hand, models the adversarial environment. In particular, he controls the spontaneous temperature changes. We make the simplifying assumption that the temperature can only change by 1 degree Fahrenheit at every time step. Moreover, if the mode is heat then the temperature cannot

$\{\mathsf{Precondition} : t = 68\}$
$\mathsf{while}\ (t = 67) \vee (t = 68) \vee (t = 69)\ \mathsf{do}$
$\quad //$ loop invariant inv:
$\quad //\quad (t = 67) \vee (t = 68) \vee (t = 69)$ and
$\quad //\quad (t = 67) \to (m = \mathsf{heat})$ and
$\quad //\quad (t = 69) \to (m = \mathsf{cool})$
$\quad \mathsf{if}\ (m = \mathsf{heat})\ \mathsf{then}\ (t := t + 1) \sqcap \mathsf{id}$
$\quad \mathsf{else\ if}\ (m = \mathsf{cool})\ \mathsf{then}\ (t := t - 1) \sqcap \mathsf{id}$
$\quad \mathsf{else\ if}\ (m = \mathsf{off})\ \mathsf{then}\ (t := t + 1) \sqcap (t := t - 1) \sqcap \mathsf{id}$
$\quad //\ (t = 67) \vee (t = 68) \vee (t = 69)$
$\quad (m := \mathsf{heat}) \sqcup (m := \mathsf{cool}) \sqcup (m := \mathsf{off})$
$\{\mathsf{Postcondition} : \mathsf{false}\}$

$\mathbf{\Phi} : (t \neq 67) \vee (t \neq 68)$
$(t \neq 67) \vee (t \neq 69)$
$(t \neq 68) \vee (t \neq 69)$
$(m = \mathsf{heat}) \vee (m = \mathsf{cool}) \vee (m = \mathsf{off})$
$(m \neq \mathsf{heat}) \vee (m \neq \mathsf{cool})$
$(m \neq \mathsf{heat}) \vee (m \neq \mathsf{off})$
$(m \neq \mathsf{cool}) \vee (m \neq \mathsf{off})$

$\mathbf{\Psi} : \{t = 67\}t := t + 1\{t = 68\}$
$\{t = 68\}t := t + 1\{t = 69\}$
$\{t = 69\}t := t + 1\{\neg ok\}$
$\{m = v\}t := t + 1\{m = v\},$ for $v = \mathsf{heat}, \mathsf{cool}, \mathsf{off}$
$\{t = 67\}t := t - 1\{\neg ok\}$
$\{t = 68\}t := t - 1\{t = 67\}$
$\{t = 69\}t := t - 1\{t = 68\}$
$\{m = v\}t := t - 1\{m = v\},$ for $v = \mathsf{heat}, \mathsf{cool}, \mathsf{off}$
$\{\mathsf{true}\}m := \mathsf{heat}\{m = \mathsf{heat}\}$
$\{\mathsf{true}\}m := \mathsf{cool}\{m = \mathsf{cool}\}$
$\{\mathsf{true}\}m := \mathsf{off}\{m = \mathsf{off}\}$
$\{t = v\}m := w\{t = v\},$
for $v = 67, 68, 69$ and $w = \mathsf{heat}, \mathsf{cool}, \mathsf{off}$
$\{\neg ok\}m := w\{\neg ok\},$ for $w = \mathsf{heat}, \mathsf{cool}, \mathsf{off}$

Figure 11: A program modelling the interaction between a temperature controller and the environment, and a Hoare specification for the acceptable temperature range.

decrease, and if the mode is cool then the temperature cannot increase. We model the behavior of the environment with the term:

$$\begin{aligned}\mathrm{demon} \triangleq\ & \mathsf{if}\ (m = \mathsf{heat})\ \mathsf{then}\ (t := t + 1) \sqcap \mathsf{id}\\ & \mathsf{else\ if}\ (m = \mathsf{cool})\ \mathsf{then}\ (t := t - 1) \sqcap \mathsf{id}\\ & \mathsf{else\ if}\ (m = \mathsf{off})\ \mathsf{then}\ (t := t + 1) \sqcap (t := t - 1) \sqcap \mathsf{id},\end{aligned}$$

where the variable t stores the current temperature. The requirement for the temperature controller is that it keeps the temperature within the range $\{67, 68, 69\}$, expressed as

$$ok \triangleq (t = 67) \vee (t = 68) \vee (t = 69),$$

assuming that the initial temperature is 68 degrees Fahrenheit (20 degrees Celsius).

In Figure 11 we see the program that describes the interaction between the controller and the environment (in discrete steps), together with a Hoare specification demanding that the temperature is within the acceptable range. The while loop keeps executing until

a violation of the temperature range occurs. In other words, the specification is satisfied when the loop keeps running forever. We assume throughout that we reason under the hypotheses Φ for atomic tests, and the hypotheses Ψ for atomic actions. The top-level steps of the proof are:

1. $(t = 68) \to inv$ [Φ, bool]
2. $inv \to ((t = 67) \wedge (m = \mathsf{heat})) \vee (t = 68) \vee ((t = 69) \wedge (m = \mathsf{cool}))$ [bool]
3. $\{(t = 67) \wedge (m = \mathsf{heat})\}\text{demon}\{ok\}$ [use Φ, Ψ]
4. $\{t = 68\}\text{demon}\{ok\}$ [use Φ, Ψ]
5. $\{(t = 69) \wedge (m = \mathsf{cool})\}\text{demon}\{ok\}$ [use Φ, Ψ]
6. $\{inv\}\text{demon}\{ok\}$ [2, 3, 4, 5]
7. $\{ok\}\text{angel}\{inv\}$ [todo]
8. $\{inv\}\text{demon}; \text{angel}\{inv\}$ [6, 7, seq]
9. $\{inv\}\mathsf{while}\ ok\ \mathsf{do}\ (\text{demon}; \text{angel})\{inv \wedge \neg ok\}$ [8, loop]
10. $inv \wedge \neg ok \to \mathsf{false}$ [bool]
11. $\{t = 68\}\mathsf{while}\ ok\ \mathsf{do}\ (\text{demon}; \text{angel})\{\mathsf{false}\}$ [1, 9, 10]

It remains to derive the assertion $\{ok\}\text{angel}\{inv\}$, which concerns the implementation of the controller.

1. $\{t = 67\}m := \mathsf{heat}\{inv\}$ [use Ψ]
2. $\{t = 67\}\text{angel}\{inv\}$ [1, ang]
3. $\{t = 69\}m := \mathsf{cool}\{inv\}$ [use Ψ]
4. $\{t = 69\}\text{angel}\{inv\}$ [3, ang]
5. $\{t = 68\}m := \mathsf{off}\{inv\}$ [use Ψ]
6. $\{t = 68\}\text{angel}\{inv\}$ [5, ang]
7. $\{(t = 69) \vee (t = 68)\}\text{angel}\{inv\}$ [4, 6, join]
8. $\{(t = 67) \vee (t = 69) \vee (t = 68)\}\text{angel}\{inv\}$ [2, 7, join]
9. $\{ok\}\text{angel}\{inv\}$ [8, bool]

If we annotate the above proof with the angelic strategies according to the synthesis calculus of Figure 10, then the implementation for the controller becomes:

$$\begin{aligned}\text{controller} \triangleq\ & \mathsf{if}\ (t = 67)\ \mathsf{then}\ m := \mathsf{heat} \\ & \mathsf{else\ if}\ (t = 69)\ \mathsf{then}\ m := \mathsf{cool} \\ & \mathsf{else}\ m := \mathsf{off}.\end{aligned}$$

We have thus established deductively that there exists an implementation satisfying the specification, and we have obtain a $\sqcup$-free program that witnesses this fact.

10. Related Work

The present paper is inspired from and builds upon the closely related line of work on the propositional fragment of Hoare logic, called *Propositional Hoare Logic* or PHL [Koz99, Koz00, CK00, KT01, Tiu02]. In [Mam14] and [Mam16], a propositional variant of Hoare

logic for mutually recursive programs is investigated. The present work differs from all this previous work in considering the combination of angelic and demonic nondeterminism, which presents significant new challenges for obtaining completeness and decision procedures.

The other line of work that largely motivated our investigations here is an extension of Propositional Dynamic Logic (PDL) [Pra76, FL77, FL79], called *Game Logic* [Par83, Par85, PP03]. This formalism was introduced more than 30 years ago in [Par83], but there are still no completeness results for full Game Logic. We stress that the theory studied here is *not* a fragment of Game Logic. Even though hypotheses-free Hoare assertions $\{p\}f\{q\}$ can be encoded in Dynamic Logic as partial correctness formulas $p \to [f]q$, there is no direct mechanism for encoding the hypotheses of an implication $\Phi, \Psi \Rightarrow \{p\}f\{q\}$ (which would correspond to some kind of global consequence relation in Dynamic Logic).

We have already discussed in the introduction that there have been proposals of semantic models with the explicit purpose of describing the interaction between angelic and demonic choices in programs: monotonic predicate transformers [BW98, Dij75, Mor98] and up-closed multirelations [Rew03, MCR04, MCR07, MC13]. We should note that the latter model of multirelations (relations from the state space S to $\wp S$ or, equivalently, functions $S \to \wp\wp S$) had appeared much earlier in the context of modal logic under the name of *neighborhood semantics* or *Scott-Montague semantics*. See [Che80] for a textbook presentation of this general semantics (called *minimal models* in [Che80]), which is useful for analyzing non-normal modal logics. These previous works study semantic objects that are related to our game functions. However, our definition of the algebra of game functions (in particular, the definition of while loops in terms of greatest fixpoints) has not been studied before. Moreover, the precise correspondence between safety games and game functions is novel.

There is an enormous amount of work on logics for the strategic interaction between agents, such as Coalition Logic, Alternating-time Temporal Logic, Strategy Logic, and many more. These logics are mostly inspired from modal and temporal logic [BdRV01], and they are typically used for reasoning about strategic ability, cooperation, agent knowledge, and so on. The recent books [vB14] and [vBGV15] contain broad surveys of the area. We know of no previous proposal, however, that offers a succinct language for describing safety games and (unconditionally) complete systems for reasoning about safety compositionally.

Coalition Logic (CL) [Pau02] is a multi-agent formalism that studies cooperation modalities $[C]$, where C is a subset of a set N of agents/players. A formula $[C]\phi$ is read as follows: "the agents C can cooperate in order to guarantee outcome ϕ". This language is sufficient for describing only very simple multi-player games consisting of finitely many steps, and it lacks a treatment of iteration.

The language of *Alternating-time Temporal Logic (ATL)* [AHK97, AHK02] includes modalities of the form $\langle\langle C\rangle\rangle$, where C is a subset of agents. The meaning of a formula $\langle\langle C\rangle\rangle\phi$ is given w.r.t. a fixed multi-player game and it says that: "the agents C have a joint strategy so that for every joint strategy of the remaining agents, the computation induced by these strategies satisfies the linear temporal property ϕ". For a fixed game, the language of ATL is sufficient for describing safety properties. ATL cannot be used, however, for the compositional description and specification of games. An ATL formula describes a global property of the entire game, where the game is fixed a priori.

Strategy Logic (SL) [CHP10] is a very powerful extension of ATL that allows explicit quantification over the strategies of the players, instead of treating the strategies implicitly using modalities. By making strategy quantification a primitive of the language, SL can describe interesting notions of non-zero-sum games such as Nash equilibria. Similarly to

ATL, SL is interpreted over a single fixed game graph. Thus, the language of SL does not offer syntax for the compositional description and analysis of complex game graphs from simpler ones.

The work of Moggi on monads and computational effects [Mog91], where concepts from category theory are used to structure the denotational semantics of programs, has inspired work on program logics that are parameterized w.r.t. a monad encapsulating the computational effects (e.g., nontermination, probabilities, nondeterminism, and so on) of the programs. Neighborhood models and related models of dual nondeterminism have been shown to give rise to monads. A generic monadic framework for weakest precondition semantics is studied in [Has15], and a relatively complete monadic Hoare logic is proposed in [GS13]. As far as we know, none of the works in this line of research provides an operationally justified semantics for dual nondeterminism nor an unconditional completeness result.

11. Discussion & Conclusion

We have considered here the weak (over the class *All*) and the strong (over the subclass *Dem*) Hoare theories of dual nondeterminism, and we have obtained sound and unconditionally complete Hoare-style calculi for both of them. We have also shown that both theories can be decided in exponential time, and that the strong Hoare theory is EXPTIME-hard. Finally, we have extended our proof system so that it constructs program terms for the strategies of the angel, thus obtaining a sound and complete calculus for synthesis.

To the best of our knowledge, the present results are the first completeness theorems for logics of while programs that support dual nondeterminism. Handling the case of iteration in the presence of both angelic and demonic nondeterminism requires a careful treatment, since we generally need transfinitely many iterations for the loop approximants. In order to gain confidence that the employed semantics is indeed meaningful, we have shown that it agrees exactly with the intended operational model (based on safety games).

There is still much progress to be made in the problem of axiomatizing Game Logic [Par83] or a reasonable variation of it (possibly using a restricted class of models and a different syntax for programs). It also remains an interesting challenge to give equational axiomatizations for dual nondeterminism and iteration in the style of Kleene algebra [Koz94] and Kleene algebra with tests [Koz97]. For practical applications such equational theories would need to accommodate additional hypotheses for the domain of computation [KM14, GKM14, Mam15a], similarly to the use of hypotheses Φ and Ψ in our calculi. We hope that the present work will inspire progress for the aforementioned and other related open problems.

Acknowledgement

The author would like to thank the anonymous referees for their very helpful comments.

References

[AHK97] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 100–109, 1997.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[Apt81] Krzysztof R. Apt. Ten years of Hoare's logic: A survey – Part I. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 3(4):431–483, 1981.

[Apt83] Krzysztof R. Apt. Ten years of Hoare's logic: A survey – Part II: Nondeterminism. *Theoretical Computer Science*, 28(1):83–109, 1983.

[BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[BvW90] Ralph-Johan R. Back and Joakim von Wright. Duality in specification languages: A lattice-theoretical approach. *Acta Informatica*, 27(7):583–625, 1990.

[BvW92] Ralph-Johan R. Back and Joakim von Wright. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science*, 100(2):365–383, 1992.

[BW98] Ralph-Johan Back and Joakim Wright. *Refinement Calculus: A Systematic Introduction*. Springer Heidelberg, 1998.

[Che80] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[CHP10] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.

[CK00] Ernie Cohen and Dexter Kozen. A note on the complexity of propositional Hoare logic. *ACM Transactions on Computational Logic*, 1(1):171–174, 2000.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

[Coo78] Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM Journal on Computing*, 7(1):70–90, 1978.

[CvW03] Orieta Celiku and Joakim von Wright. Implementing angelic nondeterminism. In *Tenth Asia-Pacific Software Engineering Conference*, pages 176–185, 2003.

[Dij75] Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

[FL77] Michael J. Fischer and Richard E. Ladner. Propositional modal logic of programs. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (STOC '77)*, pages 286–294, 1977.

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[Flo67] Robert W. Floyd. Assigning meanings to programs. In *Mathematical Aspects of Computer Science, Proceedings of AMS Symposium in Applied Mathematics*, volume 19, pages 19–32, 1967.

[GKM14] Niels Bjørn Bugge Grathwohl, Dexter Kozen, and Konstantinos Mamouras. KAT + B! In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 44:1–44:10, 2014.

[GL73] Stephen J. Garland and David C. Luckham. Program schemes, recursion schemes, and formal languages. *Journal of Computer and System Sciences*, 7(2):119–160, 1973.

[GS13] Sergey Goncharov and Lutz Schröder. A relatively complete generic Hoare logic for order-enriched effects. In *Proceedings of the 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS '13)*, pages 273–282, 2013.

[Has15] Ichiro Hasuo. Generic weakest precondition semantics from monads enriched with order. *Theoretical Computer Science*, 604:2–29, 2015.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580,583, 1969.

[KM14] Dexter Kozen and Konstantinos Mamouras. Kleene algebra with equations. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP '14)*, pages 280–292, 2014.

[Koz94] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[Koz97] Dexter Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997.

[Koz99] Dexter Kozen. On Hoare logic and Kleene algebra with tests. In *Proceedings of the 14th Symposium on Logic in Computer Science (LICS '99)*, pages 167–172, 1999.

[Koz00] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *ACM Transactions on Computational Logic*, 1(1):60–76, 2000.

[KT01] Dexter Kozen and Jerzy Tiuryn. On the completeness of propositional Hoare logic. *Information Sciences*, 139(3-4):187–195, 2001.

[LPP70] David C. Luckham, David M. R. Park, and Michael S. Paterson. On formalised computer programs. *Journal of Computer and System Sciences*, 4(3):220–249, 1970.

[Mam14] Konstantinos Mamouras. On the Hoare theory of monadic recursion schemes. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 69:1–69:10, 2014.

[Mam15a] Konstantinos Mamouras. *Extensions Of Kleene Algebra For Program Verification*. PhD thesis, Cornell University, Ithaca, NY, August 2015.

[Mam15b] Konstantinos Mamouras. Synthesis of strategies and the Hoare logic of angelic nondeterminism. In Andrew Pitts, editor, *Proceedings of the 18th International Conference on Foundations of Software Science and Computation Structures (FOSSACS '15)*, volume 9034 of *Lecture Notes in Computer Science*, pages 25–40. Springer, 2015.

[Mam16] Konstantinos Mamouras. The Hoare logic of deterministic and nondeterministic monadic recursion schemes. *ACM Transactions on Computational Logic (TOCL)*, 17(2):13:1–13:30, 2016.

[MC13] Clare E. Martin and Sharon A. Curtis. The algebra of multirelations. *Mathematical Structures in Computer Science*, 23:635–674, 2013.

[MCR04] Clare E. Martin, Sharon A. Curtis, and Ingrid Rewitzky. Modelling nondeterminism. In *Proceedings of the 7th International Conference on the Mathematics of Program Construction (MPC '04)*, pages 228–251, 2004.

[MCR07] Clare E. Martin, Sharon A. Curtis, and Ingrid Rewitzky. Modelling angelic and demonic nondeterminism with multirelations. *Science of Computer Programming*, 65(2):140–158, 2007.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[Mor98] Carroll Morgan. *Programming From Specifications*. Prentice-Hall, 1998.

[Par83] Rohit Parikh. Propositional game logic. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS '83)*, pages 195–200, 1983.

[Par85] Rohit Parikh. The logic of games and its applications. In Marek Karplnski and Jan van Leeuwen, editors, *Topics in the Theory of Computation – Selected Papers of the International Conference on Foundations of Computation Theory, FCT '83*, volume 102 of *North-Holland Mathematics Studies*, pages 111–139. North-Holland, 1985.

[Pat68] Michael S. Paterson. Program schemata. In *Machine Intelligence 3*, pages 19–31. Edinburgh University Press, 1968.

[Pau02] Marc Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[PH70] Michael S. Paterson and Carl E. Hewitt. Comparative schematology. In Jack B. Dennis, editor, *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, pages 119–127. ACM, 1970.

[PP03] Marc Pauly and Rohit Parikh. Game logic — An overview. *Studia Logica*, 75(2):165–182, 2003.

[Pra76] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 17th IEEE Annual Symposium on Foundations of Computer Science (FOCS '76)*, pages 109–121, 1976.

[Rew03] Ingrid Rewitzky. Binary multirelations. In *Theory and Applications of Relational Structures as Knowledge Instruments*, pages 256–271. Springer, 2003.

[Rut64] Joseph D. Rutledge. On Ianov's program schemata. *Journal of the ACM*, 11(1):1–9, 1964.

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS '95)*, pages 1–13, 1995.

[Tiu02] Jerzy Tiuryn. Hoare logic: From first-order to propositional formalism. In *Proof and System-Reliability*, pages 323–340. Springer, 2002.

[vB14] Johan van Benthem. *Logic in Games*. MIT Press, 2014.

[vBGV15] Johan van Benthem, Sujata Gosh, and Rineke Verbrugge, editors. *Models of Strategic Reasoning: Logics, Games, and Communities*. Springer, 2015.