Programming and Verifying Subgame Perfect Mechanisms

Marc Pauly

June 17, 2002

Abstract

An extension of the WHILE-language is developed for programming game-theoretic mechanisms involving multiple agents. Examples of such mechanisms include auctions, voting procedures, and negotiation protocols. A structured operational semantics is provided in terms of extensive games of almost perfect information. Hoare-style partial correctness assertions are proposed to reason about the correctness of these mechanisms, where correctness is interpreted as the existence of a subgame perfect equilibrium. Using an extensional approach to pre- and postconditions, we show that an extension of Hoare's original calculus is sound and complete for reasoning about subgame perfect equilibria in game-theoretic mechanisms.

1 Introduction

In recent years, games have become more prominent in different areas of computer science research. The reason for this seems to be the realisation that games form a natural generalisation of programs. This insight can be realised on a number of different levels (we shall only mention two): On a foundational level, games have been used to provide an alternative model of computation, the alternating Turning machine [3]. At a more abstract level, program logics like propositional dynamic logic have been extended to games [13].

From a game-theoretic perspective, much of this work is extremely narrow, since it mainly focuses on determined 2-player win/loss games of perfect information. On the other hand, game theory has developed a wealth of techniques to study more complicated situations where agents interact, involving more than two players, imperfect information, and preferences over outcomes which cannot be captured by simply distinguishing between winning and loosing. Still, it has been suggested [12] that combining research in game theory and computer science, we may be able to obtain a better understanding of *social software*. The present paper tries to contribute to this aim.

More concretely, we attempt to generalise techniques from formal program verification to games or game-theoretic mechanisms such as auctions, voting procedures, etc.. From a logical perspective, two approaches suggest themselves. On the one hand, one might extend model checking approaches [4], where one uses, for instance, temporal logic to specify properties of a system (program/game) and proceeds to verify these properties using model checking. This approach has been generalised to reason about coalitional power in games [14]. On the other hand, one can try to extend approaches based on theorem proving using a formal calculus in which one can derive certain properties of a system. This approach will be taken here.

The axiomatic or compositional approach to program verification was introduced by Hoare [7] and Dijkstra [5], and provided the foundation stone for formal program verification [10, 1, 6]. In Hoare's calculus, correctness assertions of the form $\{P\}\pi\{Q\}$ are used to express that program π, when executed in a state satisfying P, will terminate in a state satisfying Q. In generalising the program verification approach to games, this paper makes two contributions: First, it defines a programming language which is a simple extension of the WHILE-language sufficient to program game-theoretic mechanisms. The syntax of this language is defined in section 2, and section 3 provides a structured operational semantics in terms of extensive games of almost perfect information. Second, we are going to extend Hoare's calculus to reason about the correctness of these mechanisms, where correctness is interpreted as the existence of a subgame perfect equilibrium with a certain payoff. In section 4, we define our new notion of correctness by providing a game-theoretic interpretation of $\{P\}\pi\{Q\}$, and linking it to the game-theoretic notion of implementation and mechanism design. Finally, section 5 presents an extensional calculus for reasoning about mechanism correctness, and provides proofs of soundness and completeness.

Since the aim of this paper is to provide the theoretical foundations for programming mechanisms and reasoning about their equilibria, we decided to simplify and shorten the exposition in two ways: First, the calculus presented in section 5 is extensional rather than intensional (see discussion in section 4.1). Second, while we do provide a number of examples of concrete mechanisms and how they can be programmed, these examples are merely intended to convince the reader of the generality of the programming language defined and its applicability in different areas. We do not present any derivations of correctness assertions, and rather intend to leave those for a subsequent paper which would also include an intensional version of our calculus.

2 Syntax of MPL

Our mechanism programming language (MPL) is a simple extension of standard imperative programming languages; more concretely, our point of departure is the well-known WHILE-language (see e.g. [10]). We assume throughout that we are given a nonempty set of agents or players Ags, a set of mechanism variables MV, a set of function symbols $Funs$ and a set of relation symbols $Rels$. Using these, we inductively define terms t, boolean expressions B and mechanisms (or game forms) γ as follows:

$$
\begin{array}{rl}
t := & x \mid f^k(t_1,\ldots,t_k) \\
B := & true \mid false \mid R^k(t_1,\ldots,t_k) \mid \neg B \mid B_1 \wedge B_2 \\
\gamma := & x := t \mid \gamma_1;\gamma_2 \mid \texttt{if}\, B\, \texttt{then}\, \gamma_1\, \texttt{else}\, \gamma_2 \mid \texttt{while}\, B\, \texttt{do}\, \gamma \mid \\
& \texttt{ch}_A(\{x_a | a \in A\})
\end{array}
$$

where $a \in Ags$, $f^k \in Funs$ is a k-ary function symbol (in case $k = 0$ we are dealing with constants), $R^k \in Rels$ a k-ary relation symbol, $x, x_a \in MV$, and $A \subseteq Ags$ is finite and nonempty.

The last construct presents the only addition to the standard WHILE-language: $\texttt{ch}_A$ lets agent a choose any value for the variable x_a. The agents are making their choice simultaneously, so we require all the x_a to be distinct. One can think of the $\texttt{ch}_A$ construct as a strategic game among n agents, where the strategic choices of each agent are represented by his choice of the value of a variable. While the set of agents may be infinite, we require each $\texttt{ch}_A$ construct to involve only finitely many agents. In the special case where $|A| = 1$, we have a simple nondeterministic choice. More concretely, in case agent 1 can choose between 2 different strategies, executing γ_1 vs. executing γ_2, we can describe this situation as

$$\texttt{ch}_{\{1\}}(\{x_1\}); \texttt{if}\, x_1 = 0\, \texttt{then}\, \gamma_1\, \texttt{else}\, \gamma_2,$$

where we assume that the domain of computation is the set of natural numbers, for instance, and $= \in Rels$ and $0 \in Funs$.

MPL is an extremely general programming language for a large variety of different kinds of mechanisms. In section 4 we shall use it for defining mechanisms for different kinds of auctions. Voting procedures are further examples of mechanisms which can be programmed using MPL. As an example, the well-known Borda-count procedure (see, e.g., [2]) can be programmed as follows:

```
ch_Ags({x_1, x_2, ..., x_N});
a := 1;
i := 1;
while i ≠ K do c_i := 0; i := i + 1;
while  a ≠ N do
        i := 1;
        while  i ≠ K do
                c_i := c_i + x_a[i];
                i := i + 1;
        a := a + 1
```

In this example, we assume that $Ags = \{1, 2, \ldots, N\}$, and that the agents have to choose among K candidates. First, each agent a can cast a ballot of the form $x_a = (p_1, p_2, \ldots, p_K)$, where p_i is the number of points the agent gives to candidate i. Ballots have to be rankings of candidates, i.e., the most preferred candidate must obtain K points, the next preferred candidate $K - 1$ points, etc., so that the least preferred candidate obtains 1 point. Hence, we assume implicitly, that the domain of computation contains these possible ballots, and that the initial choice assigns a ballot to each x_a (note that since the domain of computation will also contain the natural numbers, we need to make sure that each x_a is assigned to an element of the appropriate ballot type, but we shall ignore this problem in order to keep the algorithm simple). Once the ballots are cast, a is initialised to the first agent, and i to the first candidate. The variable c_i counts the number of points accumulated by candidate i, and is initialised to 0. The main part of the algorithm then simply sums up the points for each candidate, where $x_a[i]$ refers to p_i, in case $x_a = (p_1, p_2, \ldots, p_K)$. The winner of the vote will be the candidate accumulating the most points.

A further example of a well-known mechanism which can be programmed in MPL is a version of Rubinstein's negotiation protocol of alternating offers (see [11, 8]).

$agree := false;$
$optout := false;$
$i := 1;$
$\texttt{while } \neg optout \wedge \neg agree \texttt{ do}$
$\quad \texttt{if } i = 1 \texttt{ then } \texttt{ch}_{\{1\}}(\{x\}) \texttt{else } \texttt{ch}_{\{2\}}(\{x\});$
$\quad \texttt{if } i = 1 \texttt{ then } \texttt{ch}_{\{2\}}(\{y\}) \texttt{else } \texttt{ch}_{\{1\}}(\{y\});$
$\quad \texttt{if } y = 0 \texttt{ then } agree := true$
$\qquad\quad \texttt{else if } y = 1 \texttt{ then } optout := true \texttt{ else } i := (i \, mod \, 2) + 1$

For simplicity, we have assumed that there are only two agents who try to reach an agreement over, e.g., the price of a car which agent 1 wants to sell to agent 2, and so we can assume the domain of computation to be simply the natural numbers. The negotiation procedure can end in an agreement concerning the price, one of the agents can opt out of the negotiation (in which case some predetermined event will occur), or the negotiation can go on forever. The protocol starts by agent 1 making a price offer x. Agent 2 responds by choosing y, where we interpret $y = 0$ as signalling agreement to the price offered, $y = 1$ as a decision to opt out of the negotiation, and any other value for y as signalling the desire to make a counteroffer, upon which we get another iteration of the loop with the roles reversed.

The above negotiation protocol is very general, and numerous instances of it have been analysed game-theoretically [8]. We shall not go into this or the voting mechanism in more detail, since our main aim is only to suggest the generality of the mechanism programming language defined. A more detailed and more formal treatment of examples such as the ones given will be left for the future. In the following section, we shall provide a formal semantics for this language in terms of games. Furthermore, we will provide a calculus for reasoning about the existence of game-theoretic equilibria in these mechanisms, and about the payoffs the agents obtain in equilibrium.

Note that MPL only allows one to construct mechanisms with almost-perfect information, i.e., agents are perfectly informed about all the choices made except possibly for simultaneous moves. Different subclasses of MPL-mechanisms correspond to various natural assumptions regarding the power of the mechanism designer and the agents in general. The class MPL(PRG) of *programs* is the class of MPL-mechanisms which do not contain any $\mathtt{ch}_A$ construct. Without this construct, MPL is simply the WHILE-language. The class MPL(PI) of *perfect-information mechanisms* will restrict the use of $\mathtt{ch}_A$ to cases where $|A| = 1$, i.e., where all choices involve only a single agent. Perfect-information mechanisms allow different agents to make choices at different times, but all choices are public.

3 Structured Operational Semantics via Games

The most detailed semantics we can provide for MPL expressions is a structured operational semantics which specifies the configurations a mechanism can be in and the possible transitions between configurations. For programs, such a semantics gives rise to an execution sequence or trace, and in case of nondeterministic programs to an execution tree. Since in the case of mechanism we are dealing with multiple agents, we arrive at a game tree whose positions are the possible configurations of the mechanism.

As is standard in first-order logic, we will work with an interpretation $\mathcal{I}$ which provides us with a domain $D_\mathcal{I}$ and functions and relations over $D_\mathcal{I}$ as interpretations for the symbols in $Funs$ and $Rels$. Furthermore, we assume that besides the relations associated to symbols in $Rels$, our interpretation contains an additional binary $\geq_a^\mathcal{I}$-relation for every agent $a \in Ags$. The $\geq_a^\mathcal{I}$ relation will be used to represent agent a's preference over the elements of the domain. Note that mechanisms programmed in MPL cannot refer to these preferences, since $\geq_a \notin Rels$.

The only requirements on $\mathcal{I}$ are that the preference relations $\geq_a^\mathcal{I} \subseteq D_\mathcal{I} \times D_\mathcal{I}$ satisfy the following properties: (1) $\geq_a^\mathcal{I}$ must be a partial pre-order, i.e., a reflexive and transitive relation on $D_\mathcal{I}$, and (2) there is a uniformly worst outcome (which we denote as $-\infty$), i.e., there is some $d \in D_\mathcal{I}$ such that for all $i \in Ags$ and $x \in D_\mathcal{I}$ we have $x \geq_i d$.

A state $s : MV \to D_\mathcal{I}$ is a function assigning a domain element to each mechanism variable. Let $S_\mathcal{I}$ be the set of all states over $\mathcal{I}$. In general, whenever the intended interpretation $\mathcal{I}$ is clear we shall tend to omit it. The following standard logical notation will be used: $\mathcal{I}, s \models \varphi$ denotes that a first-order formula φ whose variables are all in MV is true in $\mathcal{I}$ at state s. Similarly, we let $\varphi^\mathcal{I} = \{s \in S_\mathcal{I} | \mathcal{I}, s \models \varphi\}$. Again, when the intended interpretation is clear, we shall often simply write $s \models \varphi$.

Given interpretation $\mathcal{I}$ and an initial state s_0, we shall interpret every mechanism γ as a game form of almost-perfect information $G(\gamma, s_0, \mathcal{I})$. Let Cfg denote the set of *configurations*, i.e., the set of all pairs $\langle \gamma, s\rangle$ where γ is a mechanism or the empty mechanism Λ, and s is a state. We define a transition relation $\xrightarrow{A} \subseteq Cfg \times Cfg$ for $A \subseteq Ags$ such that $c \xrightarrow{A} c'$ states that the game can proceed from c to c' provided the agents A make some choice/move. In case the move does not require any agent to make a choice, we will have $A = \emptyset$. In the standard way (see e.g. [10]), we define the $\xrightarrow{A}$ relation inductively as the smallest sets satisfying the following axioms and inference rules, the only novelty here being the definition for $\mathtt{ch}_A$:

$$\langle x := t, s\rangle \xrightarrow{\emptyset} \langle \Lambda, s^x_t\rangle$$

$$\langle \mathtt{ch}_A(\{x_i | i \in A\}), s\rangle \xrightarrow{A} \langle \Lambda, s'\rangle \text{ where } s'(y) = s(y) \text{ for all } y \neq x_i$$

$$\frac{\langle \pi_1, s\rangle \xrightarrow{A} \langle \Lambda, s'\rangle}{\langle \pi_1; \pi_2, s\rangle \xrightarrow{A} \langle \pi_2, s'\rangle} \qquad \frac{\langle \pi_1, s\rangle \xrightarrow{A} \langle \pi_1', s'\rangle}{\langle \pi_1; \pi_2, s\rangle \xrightarrow{A} \langle \pi_1'; \pi_2, s'\rangle}$$

$$\frac{\mathcal{I}, s \models B}{\langle \mathtt{if}\ B\ \mathtt{then}\ \gamma_1\ \mathtt{else}\ \gamma_2, s\rangle \xrightarrow{\emptyset} \langle \gamma_1, s\rangle}$$

$$\frac{\mathcal{I}, s \not\models B}{\langle \mathtt{if}\ B\ \mathtt{then}\ \gamma_1\ \mathtt{else}\ \gamma_2, s\rangle \xrightarrow{\emptyset} \langle \gamma_2, s\rangle}$$

$$\frac{\mathcal{I}, s \not\models B}{\langle \mathtt{while}\ B\ \mathtt{do}\ \gamma, s\rangle \xrightarrow{\emptyset} \langle \Lambda, s\rangle} \qquad \frac{\mathcal{I}, s \models B}{\langle \mathtt{while}\ B\ \mathtt{do}\ \gamma, s\rangle \xrightarrow{\emptyset} \langle \gamma; \mathtt{while}\ B\ \mathtt{do}\ \gamma, s\rangle}$$

where $s^x_t(y) = s(y)$ for $y \neq x$ and $s^x_t(x) = t^{\mathcal{I},s}$, the interpretation of t in $\mathcal{I}$ at s.

Let Cfg^* be the set of all finite nonempty sequences of configurations $c_0, c_1, \ldots, c_n$ such that $c_i = \langle \gamma_i, s_i\rangle$ and

$$\langle \gamma_0, s_0\rangle \xrightarrow{A_1} \langle \gamma_1, s_1\rangle \xrightarrow{A_2} \ldots \xrightarrow{A_n} \langle \gamma_n, s_n\rangle,$$

and let Cfg^*_a be those sequences which end in a configuration c_n for which there is some configuration c_{n+1} and set $A \subseteq Ags$ such that $c_n \xrightarrow{A} c_{n+1}$ and $a \in A$. Infinite configuration sequences as well as finite configuration sequences $c_0, \ldots, c_n$ for which there is no c_{n+1} and A such that $c_n \xrightarrow{A} c_{n+1}$ are called *terminal*, and we denote the set of terminal sequences as Cfg^t.

The move relations give rise to the game tree or *semi-game* $G(\gamma, s_0, \mathcal{I})$ which starts at the initial position/configuration $\langle \gamma, s_0\rangle$. We interpret Cfg^* as the set of (partial) histories of the game, where each agent a gets to move at the positions which are in Cfg^*_a. Note that we talk of a tree, since we can think of possible loops as infinite branches. While we shall usually refer to $G(\gamma, s_0, \mathcal{I})$ as a game (omitting the "semi"), note that a semi-game lacks a link between runs/histories and preferences, for although $\mathcal{I}$ does contain information about the players' preferences over outcomes, the triple G does not have any mapping between histories of the game and outcomes. Such a mapping $\widehat{o}$ will be added shortly.

A *strategy* for agent a in game $G(\gamma_0, s_0, \mathcal{I})$ is a function $\sigma^a : Cfg^*_a \to D_{\mathcal{I}}$. Given a strategy profile $\sigma = (\sigma^1, \ldots, \sigma^n)$, i.e., a strategy σ^a for every agent

$a \in Ags$, we obtain a unique (possibly infinite) *run* which we denote as $run(\sigma)$, i.e., a maximal sequence of configurations

$$\langle \gamma_0, s_0 \rangle \xrightarrow{A_1} \langle \gamma_1, s_1 \rangle \xrightarrow{A_2} \dots$$

where $\langle \gamma_0, s_0 \rangle$ is the initial configuration, and for all $A_{k+1} \neq \emptyset$ we have $s_{k+1}(x_i) = \sigma^i(\langle \gamma_0, s_0 \rangle, \dots, \langle \gamma_k, s_k \rangle)$ for all $i \in A$, and $s_{k+1}(y) = s_k(y)$ otherwise. If $run(\sigma)$ is finite, we let s_σ denote the state associated to the terminal configuration of $run(\sigma)$.

Preferences, Predicates, and Strategic Equilibria

Each agent has certain preferences over the various possible outcomes of the mechanism. Given interpretation $\mathcal{I}$ and two outcomes $o, o' \in D_{\mathcal{I}}$, agent i prefers o at least as much as o' whenever $o \geq_i^{\mathcal{I}} o'$ holds. Given a state $s : MV \to D_{\mathcal{I}}$ and an outcome $o \in D_{\mathcal{I}}$, we call (s, o) an *extended state*, or *e-state* for short. Often, the elements in $D_{\mathcal{I}}$ will be elements of some product space, so that, e.g., $(o_1, o_2) \in \mathbb{R} \times \mathbb{R}$ will yield outcome o_1 for player 1 and outcome o_2 for player 2, where $(o_1, o_2) \geq_i (o'_1, o'_2)$ iff $o_i \geq o'_i$.

An *outcome function* $\widehat{o} : Cfg^t \to D_{\mathcal{I}}$ assigns an outcome to every terminal history, and we let $\widehat{O}$ denote the the set of all outcome functions. Given a semi-game $G(\gamma, s, \mathcal{I})$ we then obtain a *game* $G(\gamma, s, \mathcal{I}, \widehat{o})$, where for each terminal sequence of configurations $\bar{c}$ the associated outcome is $\widehat{o}(\bar{c})$, and agent i prefers $\bar{c}_1$ to $\bar{c}_2$ iff $\widehat{o}(\bar{c}_1) \geq_i \widehat{o}(\bar{c}_2)$. Given profile σ, we usually write $\widehat{o}(\sigma)$ instead of $\widehat{o}(run(\sigma))$, as we shall not be very careful about distinguishing σ from $run(\sigma)$.

Subgames of games will play a special role in the equilibrium notion to be defined subsequently. A game $G'(\gamma', s', \mathcal{I}, \widehat{o})$ is a *subgame* of a game $G(\gamma, s, \mathcal{I}, \widehat{o})$ iff there is a finite sequence of configurations $\langle \gamma_0, s_0 \rangle \xrightarrow{A_1} \langle \gamma_1, s_1 \rangle \xrightarrow{A_2} \dots \xrightarrow{A_n} \langle \gamma_n, s_n \rangle$ for some $n \geq 0$ such that $\langle \gamma_0, s_0 \rangle = \langle \gamma, s \rangle$ and $\langle \gamma_n, s_n \rangle = \langle \gamma', s' \rangle$. Given a strategy profile σ for G, we let $\sigma | G'$ denote its restriction to G', where $\sigma^a | G'(\langle \gamma_n, s_n \rangle, \dots, \langle \gamma_{n+k}, s_{n+k} \rangle) = \sigma^a(\langle \gamma_0, s_0, \rangle, \dots, \langle \gamma_n, s_n \rangle, \dots, \langle \gamma_{n+k}, s_{n+k} \rangle)$.

A *predicate* on $\mathcal{I}$ is simply a set of states $P \subseteq S_{\mathcal{I}}$, and hence every FOL formula φ containing only variables of MV gives rise to a predicate $\varphi^{\mathcal{I}}$. Similarly, an *extended predicate*, or *e-predicate* for short, is a set of e-states $P \subseteq S_{\mathcal{I}} \times D_{\mathcal{I}}$, and every FOL formula which contains variables of MV plus a new outcome variable $x_o \notin MV$ gives rise to an e-predicate. Given two predicates (or alternatively, two e-predicates), intersection, complementation, etc. can be defined simply set-theoretically. Given a predicate P_1 and an e-predicate P_2, however, we define $P_1 \cap P_2 = \{(s, o) \in P_2 | s \in P_1\}$.

Games can also be obtained from extended predicates. Given an e-predicate Q, let

$$\widehat{O}_Q = \{\widehat{o} \in \widehat{O} \mid \forall run(\sigma) \in Cfg^t : \text{ if } run(\sigma) \text{ is finite then } (s_\sigma, \widehat{o}(\sigma)) \in Q\}$$

contain all the outcome functions $\widehat{o}$ which yield an outcome satisfying Q at every finite history. Note that in general, $\widehat{O}_Q$ may be empty or contain multiple outcome assignments. But given predicate Q and some $\widehat{o}_Q \in \widehat{O}_Q$, we are able to turn a semi-game $G(\gamma, s, I)$ into a game $G(\gamma, s, \mathcal{I}, \widehat{o}_Q)$.

Now that we have defined how executions of mechanisms give rise to game trees, we can apply two well-known equilibrium notions from game theory (see, e.g., [11] for a discussion of these notions). Given a strategy profile $\sigma = (\sigma^1, \ldots, \sigma^n)$ and a strategy τ^i for player i, let (τ^i, σ^{-i}) denote the modified strategy profile $(\sigma^1, \ldots, \sigma^{i-1}, \tau^i, \sigma^{i+1}, \ldots, \sigma^n)$. Furthermore, let $\sigma{\sim_i}\tau$ denote that the strategy profiles σ and τ differ at most regarding the action prescribed for player i.

Considering any game $G(\gamma, s, \mathcal{I}, \widehat{o})$, we call a strategy profile σ a *Nash equilibrium* (NE) in G iff for all agents i and strategies τ^i we have $\widehat{o}(\sigma) \geq_i^{\mathcal{I}} \widehat{o}((\tau^i, \sigma^{-i}))$. Furthermore, σ is a *subgame perfect equilibrium* (SPE) iff for every subgame G' of G, $\sigma|G'$ is a Nash equilibrium in G'.

4 Mechanism Correctness

4.1 Hoare Logic: From Programs to Games

Hoare in [7] introduced correctness assertions of the form $\{P\}\gamma\{Q\}$, where γ is a program and P and Q are predicates. The intended interpretation of this assertion is that in every state which satisfies P, any terminating execution of program γ ends in a state which satisfies Q. In this paper, we shall extend this approach to reason about the correctness of game-theoretic mechanisms under subgame perfect equilibria.

In lifting standard Hoare triples to games we generalise them in two ways. Note that we can view the postcondition Q as specifying the winning condition for the game, i.e., all plays of the game ending in a state which satisfies Q are a win, all others a loss. Note that under the partial correctness reading, infinite runs are in fact also treated as wins. Our first generalisation consists of moving from simple win/loss situations, represented by predicates, to general preference structures. This is achieved by moving from predicates to e-predicates which also specify the outcome or payoff at a state. Second and more importantly, we move from simple claims about the existence of a strategy profile satisfying the postcondition to more refined claims about the existence of a strategy profile which has an equilibrium property. This equilibrium property is generally quite complex, and it is the complexity of this equilibrium property which can present a challenge to compositionality, in particular to the Hoare inference rule for composing two programs/games (see lemma 1 below).

Before defining our mechanism correctness assertions $\{P\}\gamma\{Q\}$, it is important to point out that we are following an *extensional* rather than an *intensional* approach (see also [10]). We assume that pre- and postconditions are predicates, i.e., semantic objects rather than formulas of some logical language. Naturally, this means that the calculus we present later is not fully syntactic. In the intensional approach, however, one runs into the problem of *expressiveness*, since it may happen that under a given interpretation the logical language is not rich enough to express all the preconditions needed. This complicates completeness proofs considerably, due to the need for an arithmetisation of syntax (Gödelisation), etc.. Furthermore, we feel that this extra work yields more insights about the logic used for the assertion language (usually first-order logic) than about the game theoretic mechanisms and their equilibria, which is what we are interested in here.

Due to its fully syntactic nature, it does seem likely that the automated verification of mechanisms would benefit from using the intensional approach, and we do intend to investigate this approach in the future (see also comments in the last section). However, note that in contrast to most computer programs whose domain of computation contains at least the natural numbers, mechanisms like voting procedures are often using a finite domain of computation, e.g., because there is only a small number of possible candidates running for president. In such cases, it may in fact be easier to do automatic verification using the extensions of the predicates directly. Second, even if this is not the case, the best logic to choose for automated verification may very much depend on the class of mechanisms under consideration, the theorem prover to be used, etc.. Hence, for our present purposes, we decide to regard these issues of more practical implementation concern, which the extensional approach allows us to avoid and postpone for further study.

4.2 Mechanism Correctness and Implementation

Assume that we are given some interpretation $\mathcal{I}$, a mechanism γ, and e-predicates P and Q. Then we say that $\{P\}\gamma\{Q\}$ is *valid* in $\mathcal{I}$, denoted as $\mathcal{I} \models \{P\}\gamma\{Q\}$, iff

> for every $(s, o) \in P$, there is an outcome function $\widehat{o} \in \widehat{O}_Q$ and a strategy profile σ such that σ is an SPE in $G(\gamma, s, \mathcal{I}, \widehat{o})$ and $\widehat{o}(\sigma) = o$.

The notion defined indeed generalises the standard partial correctness assertions of Hoare in the following way: Given an arbitrary element $d \in D_{\mathcal{I}}$ and a predicate $P \subseteq S_{\mathcal{I}}$, let $P^* = \{(s, d) | s \in P\}$. Then given any program $\gamma \in$ MPL(PRG) and predicates P and Q, the partial correctness assertion $\{P\}\gamma\{Q\}$ holds in interpretation $\mathcal{I}$ iff $\mathcal{I} \models \{P^*\}\gamma\{Q^*\}$.

In order to link our mechanism correctness assertion to the game-theoretic literature on mechanism design and implementation theory [11, 9, 15], we shall define our version of the mechanism design problem more formally. Given a set of possible outcomes $D_{\mathcal{I}}$ of the mechanism and the set of preference profiles over $D_{\mathcal{I}}$, a *social choice correspondence* f maps a preference profile $(\geq_i)_{i \in Ags}$ to a set of outcomes $X \subseteq D_{\mathcal{I}}$. The idea is that at preference profile $(\geq_i)_{i \in Ags}$, society or the mechanism designer wants one of the outcomes in $f((\geq_i)_{i \in Ags})$ to be implemented or achieved. In case $f((\geq_i)_{i \in Ags})$ is empty, society is indifferent to the outcome actually realised. The mechanism design problem is to find a mechanism which implements the social choice correspondence in a non-centralised manner, i.e., no matter what the preferences of the agents are, self-interested agents will have an incentive to play so that the outcome intended by the designer will obtain. We shall now see how this problem can be translated into our mechanism correctness assertions.

For a preference profile $(\geq_i)_{i\in Ags}$ where each $\geq_i \subseteq D_{\mathcal{I}} \times D_{\mathcal{I}}$, let $\mathcal{I}[(\geq_i)_{i\in Ags}]$ denote the model which is obtained from replacing the interpretation of the preference relations by the $\geq_i$. Furthermore, for a given social choice correspondence f, let $f^*(x) = \{(s,o) \in S_{\mathcal{I}} \times D_{\mathcal{I}} | o \in f(x)\}$, and let Q be any e-predicate. Then we say that the pair (γ, Q) *SPE-implements* a social choice correspondence f iff for all preference profiles $(\geq_i)_{i\in Ags}$ we have

$$\mathcal{I}[(\geq_i)_{i\in Ags}] \models \{f^*((\geq_i)_{i\in Ags})\}\gamma\{Q\}.$$

To see what this statement actually expresses, let us unpack the definition: (γ, Q) SPE-implements social choice correspondence f iff

> for all preference profiles $(\geq_i)_{i\in Ags}$, for all states $s \in S_{\mathcal{I}}$, and for all $o \in f((\geq_i)_{i\in Ags})$, there is some $\widehat{o} \in \widehat{O}_Q$ and some strategy profile σ such that σ is an SPE for $G(\gamma, s, \mathcal{I}[(\geq_i)_{i\in Ags}], \widehat{o})$ such that $\widehat{o}(\sigma) = o$.

Note that this notion of implementation is a weak notion which does not ask every but only some equilibrium profile to yield the desired outcome, hence strictly speaking we are dealing with mechanism design rather than implementation theory. In the remainder of this section, we shall look at a few concrete examples of mechanism design.

4.3 Auctions

Over the domain of nonnegative real numbers, the mechanism

$$\mathtt{ch}_{\{1,2\}}(\{x_1, x_2\})$$

can represent a sealed-bid auction where the two players simultaneously choose their bids. Since this game is atomic, the notions of SPE and NE coincide, and hence we can phrase the existence of Nash equilibria using the correctness notion defined earlier.

Consider first the case of a second-price auction where the player who makes the highest bid has to pay the price of the looser's bid. We assume that our model $\mathcal{I}$ has the nonnegative real numbers as its domain, and contains two constants v_1 and v_2 whose values denote the private valuations of the players. Instead of representing outcomes as pairs $o = (o_1, o_2)$ we shall assume that there are two outcome variables o_1 and o_2 which determine the payoff of player 1 and 2, respectively. The preference ordering over elements of the domain is the obvious one: $d_1 \geq_i d_2$ iff $d_1 \geq d_2$. The postcondition of this auction is the e-predicate expressed by the following formula ψ:

$$(x_1 \geq x_2 \to (o_1 = v_1 - x_2 \wedge o_2 = 0)) \wedge (x_1 < x_2 \to (o_1 = 0 \wedge o_2 = v_2 - x_1))$$

It is easy to see that this postcondition expresses the payoffs of the players in the second-price auction. Now consider the e-predicate expressed by the following formula φ:

$$(v_1 \geq v_2 \rightarrow (o_1 = v_1 - v_2 \wedge o_2 = 0)) \wedge (v_1 < v_2 \rightarrow (o_1 = 0 \wedge o_2 = v_2 - v_1))$$

Now we claim that $\mathcal{I} \models \{\varphi^{\mathcal{I}}\}\mathtt{ch}_{\{1,2\}}(\{x_1, x_2\})\{\psi^{\mathcal{I}}\}$: If player 1's valuation is at least as high as player 2's valuation, then the auction has a Nash-equilibrium in which player 2's payoff is 0 and player 1's payoff is the difference between the valuations. Similarly in case player 2's valuation is higher.

To see why this is so, note that it is a well-known result in game theory (see, e.g., [11]) that in a second-price sealed-bid auction, bidding your valuation results in a Nash equilibrium (in fact, it is even a dominant strategy). Hence, if player 1's valuation exceeds player 2's valuation, and if each player bids $x_i = v_i$, the outcomes are the ones specified by φ, and the strategies are in equilibrium. Note that the special property of this equilibrium, namely that players bid their true valuations, cannot be recovered from our formalisation.

Another interesting fact about second-price auctions is the existence of inefficient equilibria. Formally, let φ' be the e-predicate corresponding to the formula $v_1 > v_2 > 0 \wedge o_1 = 0$. Then it is the case that $\mathcal{I} \models \{\varphi'^{\mathcal{I}}\}\mathtt{ch}_{\{1,2\}}(\{x_1, x_2\})\{\psi^{\mathcal{I}}\}$: In a model in which player 1 values the object more than player 2, there is an equilibrium in which player 2 gets the object (i.e., player 1 gets payoff 0).

For first-price auctions where the winner has to pay his own bid, the earlier postcondition must be replaced by the following formula ψ':

$$(x_1 \geq x_2 \rightarrow (o_1 = v_1 - x_1 \wedge o_2 = 0)) \wedge (x_1 < x_2 \rightarrow (o_1 = 0 \wedge o_2 = v_2 - x_2))$$

In fact, the sealed-bid first-price auction is equivalent to the so-called Dutch auction, where the auctioneer continues to lower the price of an object until a player decides to take the object for the current price. Over the domain of natural numbers, the Dutch auction is captured by the following mechanism:

p:=init;
winner:=0;
$\mathtt{while}\ p \neq 0 \wedge \mathit{winner} = 0\ \mathtt{do}$
$\quad \mathtt{ch}_{\{1,2\}}(\{x_1, x_2\});$
$\quad \mathtt{if}\ x_1 > 0\ \mathtt{then}\ \mathit{winner} := 1$
$\quad\quad \mathtt{else\ if}\ x_2 > 0\ \mathtt{then}\ \mathit{winner} := 2;$
$\quad\quad\quad \mathtt{else}\ p := p - 1$

Variable p keeps track of the current price, and is initialised to some value *init*. For each offer, both players can choose a nonnegative number signaling their desire to buy the object for price p. As the algorithm is written down here, in case both players want to buy the object, player 1 gets it. Note that it is also subtleties like these which provide an argument for formally specifying and verifying mechanisms. The postcondition of the Dutch auction is the e-predicate expressed by the following formula ψ'':

$$\begin{array}{ll} & (winner = 1 \rightarrow (o_1 = v_1 - x \wedge o_2 = 0)) \\ \wedge & (winner = 2 \rightarrow (o_1 = 0 \wedge o_2 = v_2 - x)) \\ \wedge & (winner = 0 \rightarrow (o_1 = 0 \wedge o_2 = 0)) \end{array}$$

4.4 Solomon's Dilemma

The biblical dilemma of Solomon (1 Kings 3:16-28) has often been used to illustrate the basic idea of implementation theory [11, 9]. In the same spirit, we shall use it here to illustrate our notion of SPE-implementation. The game-theorist will get the additional benefit of seeing a well-known example of implementation theory translated into our framework. Solomon's dilemma is that two prostitutes have come before him with a small child, both claiming to be the mother of the child.

> He sent for a sword, and when it was brought, he said, "Cut the living child in two and give each woman half of it." The real mother, her heart full of love for her son, said to the king, "Please, Your Majesty, don't kill the child! Give it to her!" But the other woman said, "Don't give it to either of us; go on and cut it in two." Then Solomon said, "Don't kill the child! Give it to the first woman, she is its real mother."

The story exemplifies the need for a mechanism very well: Since Solomon does not know who the real mother is (i.e., he does not know the women's preferences), he cannot impose the outcome of his choice function directly. Rather, he needs to devise a mechanism which will provide an incentive to the women to reveal this information to him.

To mathematically model Solomon's situation, we consider three outcomes: a (baby is given to Anne, player 1), b (baby is given to Bess, player 2), and c (baby is cut in two). Solomon has to consider two possible situations: In case Anne is the real mother, the preference profile is given by θ_1, in case Bess is the real mother, the preference profile is θ_2.

$$\begin{array}{llcl} \theta_1 : & a >_1 b >_1 c & \text{and} & b >_2 c >_2 a \\ \theta_2 : & a >_1 c >_1 b & \text{and} & b >_2 a >_2 c \end{array}$$

Solomon's problem is to find a mechanism which implements the social choice correspondence f for which $f(\theta_1) = \{a\}$ and $f(\theta_2) = \{b\}$. In spite of Solomon's apparent cleverness, it turns out that f is not Nash-implementable. However, by slightly modifying the problem, one can obtain an implementation nonetheless.

Let us consider the situation where instead of quarrelling about a child, Anne and Bess argue about who is the owner of a painting. Furthermore, we allow Solomon to impose fines on the two women, i.e., we allow for monetary sidepayments. We can then think of the possible outcomes as triples (x, m_1, m_2), where $x \in \{0,1,2\}$ denotes who obtains the painting (0 denoting that it is cut in two), and m_i denotes the fine player i has to pay to Solomon. Now suppose that the legitimate owner of the paining has valuation v_H and the other woman has valuation v_L, where $v_H > v_L > 0$. Then if player i does not get the painting, her payoff is $-m_i$. If she does get the painting, her payoff will be $v_H - m_i$ in case she is the legitimate owner, and $v_L - m_i$ otherwise. If player i is the legitimate owner, these payoffs will then induce a preference profile θ_i in the obvious way. In this new setup, Solomon wishes to implement the social choice rule f for which $f(\theta_i) = \{(i,0,0)\}$, i.e., the painting is given to the legitimate owner and nobody has to pay any fines (we assume here that Solomon does not engage in dispute resolution to make money). More precisely, Solomon is looking for a pair (γ, Q) which SPE-implements f, i.e., for which

$$\mathcal{I}[\theta_1] \models \{o = (1,0,0)\}\gamma\{Q\} \text{ and } \mathcal{I}[\theta_2] \models \{o = (2,0,0)\}\gamma\{Q\}.$$

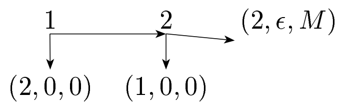
The following mechanism γ achieves this goal: First, Anne is asked whether the painting is hers or not. If she says no, the painting is given to Bess and no fines are imposed. Otherwise, Bess is asked the same question. If Bess answers the painting is not hers, it is given to Anne, again without imposing any fines. Finally, in case both players have claimed to be the owner of the painting, the painting is cut in two, Anne is fined a small amount $\epsilon > 0$ and Bess has to pay a large amount M for which $v_L < M < v_H$. The mechanism γ can be programmed as follows, where we take the real numbers as our domain:

```
ch{1}({x1});
if x1 > 0  then owner := 2
           else ch{2}({x2});
                if x2 > 0 then owner := 1 else owner := 0
```

As for the payoff specification, let Q be the e-predicate corresponding to the following formula:

$$\begin{array}{ll} & (owner = 1 \to o = (1,0,0)) \\ \wedge & (owner = 2 \to o = (2,0,0)) \\ \wedge & (owner = 0 \to o = (2,\epsilon,M)) \end{array}$$

Game theoretically, it is easy to verify that for preference profile θ_i, the following game form has a subgame perfect equilibrium yielding outcome $(i,0,0)$.



5 Axiomatic Mechanism Verification

5.1 A Hoare-style Calculus

Below we present a calculus for deriving the correctness assertions we introduced above. Note that the calculus is a natural generalisation of the standard Hoare calculus, where the only addition is an axiom for the new construct $\mathtt{ch}_A$. Given e-predicate P, we let $P[x/t] = \{(s,o) \in S_{\mathcal{I}} \times D_{\mathcal{I}} | (s_t^x, o) \in P\}$.

$$\{Q[x/t]\}x := t\{Q\} \qquad \text{(ass.)}$$

$$\{Eq(A,Q)\}ch_A(\{x_i | i \in A\})\{Q\} \qquad \text{(choice)}$$

$$\frac{\{P\}\gamma_1\{R\} \quad \{R\}\gamma_2\{Q\}}{\{P\}\gamma_1;\gamma_2\{Q\}} \qquad \text{(comp.)}$$

$$\frac{\{P \cap B^{\mathcal{I}}\}\gamma_1\{Q\} \quad \{P \cap \overline{B^{\mathcal{I}}}\}\gamma_2\{Q\}}{\{P\}\mathtt{if}\, B\, \mathtt{then}\, \gamma_1\, \mathtt{else}\, \gamma_2\{Q\}} \qquad \text{(if)}$$

$$\frac{\{P \cap B^{\mathcal{I}}\}\gamma\{P\}}{\{P\}\mathtt{while}\, B\, \mathtt{do}\, \gamma\{P \cap \overline{B^{\mathcal{I}}}\}} \qquad \text{(while)}$$

$$\frac{P \subseteq P',\ \{P'\}\gamma\{Q'\},\ Q' \subseteq Q}{\{P\}\gamma\{Q\}} \qquad \text{(l.c.)}$$

where $Eq(A,Q) = \{(s,o) \in S_{\mathcal{I}} \times D_{\mathcal{I}} | \exists \widehat{o} \in \widehat{O}_Q \exists \sigma : \sigma$ is an SPE in $G(ch_A(\{x_i | i \in A\}), s, \mathcal{I}, \widehat{o})$ and $\widehat{o}(\sigma) = o\}$.

Let $\Delta_{\mathcal{I}}$ be the smallest set of correctness assertions $\{P\}\gamma\{Q\}$ over $\mathcal{I}$ which includes the axioms and is closed under the inference rules above. We shall usually write $\{P\}\gamma\{Q\} \in \Delta_{\mathcal{I}}$ as $\mathcal{I} \vdash \{P\}\gamma\{Q\}$.

5.2 Soundness

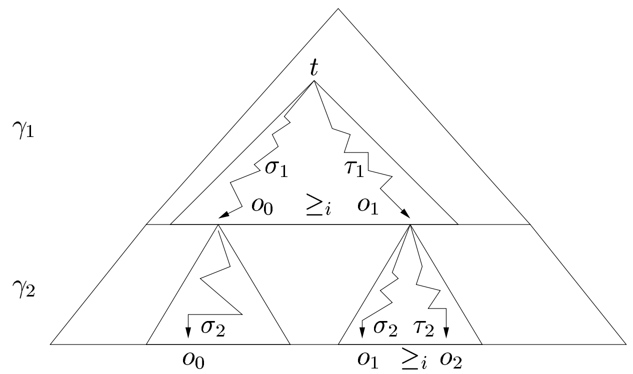
The following lemma presents the first of the two most difficult cases of the subsequent soundness result. It guarantees that equilibria of subgames can be composed into equilibria of the supergame.

Lemma 1 (Composition) *If we have both* $\mathcal{I} \models \{P\}\gamma_1\{R\}$ *and* $\mathcal{I} \models \{R\}\gamma_2\{Q\}$ *then* $\mathcal{I} \models \{P\}\gamma_1;\gamma_2\{Q\}$.

Proof. Let $(s,o) \in P$, and consider $G(\gamma_1;\gamma_2, s, \mathcal{I})$. By our first assumption, there is an outcome function $\widehat{o}_1 \in \widehat{O}_R$ and a strategy profile σ_1 such that σ_1 is an SPE in $G_1(\gamma_1, s, \mathcal{I}, \widehat{o}_1)$ and $\widehat{o}_1(\sigma_1) = o$.

Now for every finite run τ_1 of G_1 ending in some terminal state t with $\widehat{o}_1(\tau_1) = o_t$, since $(t, o_t) \in R$, we know by our second assumption that there is some outcome function $\widehat{o}_t \in \widehat{O}_Q$ and some strategy profile σ_t such that σ_t is an SPE in $G_t(\gamma_2, t, \mathcal{I}, \widehat{o}_t)$ and $\widehat{o}_t(\sigma_t) = o_t$. Taken together, σ_1 and the σ_t induce a strategy profile σ for G, and similarly $\widehat{o}_1$ (for the infinite runs of G_1) and the $\widehat{o}_t$ induce an outcome function $\widehat{o} \in \widehat{O}_Q$ for G. Hence, it remains to show that σ has the two required properties.

First, it is easily seen that $\widehat{o}(\sigma) = o$, for $\widehat{o}_1(\sigma_1) = o$, and so in case σ_1 is finite, $(s_{\sigma_1}, o) \in R$, from which by definition it follows that $\widehat{o}(\sigma) = o$. Second, we need to show that σ is an SPE in $G(\gamma_1;\gamma_2, s, \mathcal{I}, \widehat{o})$. So consider any subgame $G'(\pi, t, \mathcal{I})$ of G. In the easy case, G' will be a subgame of some $G_{t'}$, where t' is a terminal state in G_1, for in this case, our second assumption immediately guarantees the equilibrium property. In the more complicated case, G' lies partly in G_1. For simplicity, we shall for the rest of this argument assume that $\sigma = \sigma_1 \cdot \sigma_2$ refers to its restriction to G'. So consider any strategy profile $\tau_1 \cdot \tau_2$ for G' such that $\sigma = \sigma_1 \cdot \sigma_2 \sim_i \tau_1 \cdot \tau_2 = \tau$, where σ_1 and τ_1 both yield finite runs. Suppose further that $\widehat{o}(\sigma) = o_0$ and $\widehat{o}(\tau) = o_2$, as depicted below.



Now supposing that $\widehat{o}(\tau_1 \cdot \sigma_2) = o_1$, we know by definition of σ that $o_1 \geq_i o_2$, and that $\widehat{o}_1(\tau_1) = o_1$. Furthermore, since σ_1 was an SPE in G_1, we know also that $\widehat{o}_1(\sigma_1) \geq_i o_1$. Since $o_0 = \widehat{o}(\sigma) = \widehat{o}_1(\sigma_1)$, we can conclude by transitivity that $o_0 \geq_i o_2$.

Finally, note that the case where either σ_1 or τ_1 or both are infinite can be treated by a simplification of the above argument. $\square$

The following lemma isolates the arguments needed to prove the soundness of the inference rule for iteration. Our assumption that our model $\mathcal{I}$ contains a uniformly worst element is needed here.

Lemma 2 *If* $\mathcal{I} \models \{P \cap B^{\mathcal{I}}\}\gamma\{P\}$ *then* $\mathcal{I} \models \{P\}\mathtt{while}\, B\, \mathtt{do}\, \gamma\{P \cap \overline{B^{\mathcal{I}}}\}$.

Proof. Roughly speaking, the proof is an iterated application of the preceding composition lemma, but a few subtleties have to be dealt with, in particular the possibility of newly arising infinite runs.

Suppose that $(s, o) \in P$. In order to define a strategy σ and outcome function $\widehat{o}$ for $G(\mathtt{while}\, B\, \mathtt{do}\, \gamma, s, \mathcal{I})$, we shall inductively define strategy profile σ_n and outcome function $\widehat{o}_n$ for game G_n which consists of the first n iterations of game G. Game G_0 simply consists of configuration (Λ, s), strategy profile σ_0 consists of doing nothing, and as an outcome function we take $\widehat{o}_1((\Lambda, s)) = o$. Note that $\widehat{o}_1 \in \widehat{O}_P$.

For the inductive step, define G_{n+1} as G_n where for every terminal state $(t, o_t) \in P \cap B^{\mathcal{I}}$ in G_n we concatenate $G_t(\gamma, t, \mathcal{I})$ to t. By our assumption, for each such terminal state, we have an outcome function $\widehat{o}_t$ and a strategy profile σ_t, and we define σ_{n+1} and $\widehat{o}_{n+1}$ in the natural way, by extending σ_n and $\widehat{o}_n$ to G_{n+1} using the $\widehat{o}_t$ and σ_t.

Now with a slight abuse of notation, we can define strategy profile σ and outcome function $\widehat{o}$ for G as follows: We take $\sigma = \bigcup_i \sigma_i$, i.e., we simply take the profile generated by the σ_i. Similarly, we define $\widehat{o} = \bigcup_i \widehat{o}_i$, i.e., every run τ of G which is part of some G_i is evaluated according to $\widehat{o}_i$. Furthermore, there may be new infinite runs in G which are not part of any G_i, but are instead generated by an infinite number of plays of γ itself. Given such an infinite run τ, we define $\widehat{o}(\tau) = o_c$ in case there is some j such that for all $k \geq j$ we have $\widehat{o}_k(\tau|G_k) = o_c$; otherwise, we let $\widehat{o}(\tau) = -\infty$. Thus, for infinite runs which converge on a certain outcome o_c, we assign o_c to the run, and otherwise simply the uniformly worst outcome. Note that $\widehat{o} \in \widehat{O}_{P\cap\overline{B^{\mathcal{I}}}}$.

Observe first that $\widehat{o}(\sigma) = o$. For we have $\widehat{o}_1(\sigma_1) = o$, $\widehat{o}_2(\sigma_2) = \widehat{o}_1(\sigma_1) = o$, etc., and so in case σ is finite, there is some maximal k such that $\widehat{o}(\sigma) = \widehat{o}_k(\sigma_k) = o$. In case σ is infinite, we have a constant and hence converging sequence of outcomes consisting of o only.

Hence, all we need to show is that σ is an SPE in $G(\texttt{while}\, B\, \texttt{do}\, \gamma, s, \mathcal{I}, \widehat{o})$. So consider any subgame G' of G and a strategy $\tau \sim_i \sigma$ such that $\widehat{o}(\sigma) = o_0$ and $\widehat{o}(\tau) = o_2$. Now the reasoning can proceed along the lines of the composition lemma and the figure given there: In case τ yields a run which lies in G_k, we can show by induction on k that $o_0 \geq_i o_2$, each step involving the reasoning carried out in the composition lemma. On the other hand, in case τ is an infinite run generated by infinitely many γ-repetitions, we need to distinguish two cases: In the easy case where $\widehat{o}(\tau) = -\infty$, the result is obvious. In the more complicated case, $\widehat{o}(\tau) = o_c$ due to a sequence of outcomes which converges on o_c. Suppose k is the smallest number for which $\widehat{o}_k(\tau|G_k) = o_c$. Then again we can apply the reasoning of the composition lemma k times to show that $o_0 \geq_i o_c$. □

Theorem 3 (Soundness) *If* $\mathcal{I} \vdash \{P\}\gamma\{Q\}$ *then* $\mathcal{I} \models \{P\}\gamma\{Q\}$.

Proof. The proof is by induction on the length of the derivation, so we start with showing the validity of the axioms. The soundness of the $\texttt{ch}_A$ axiom follows by definition.

For $\mathcal{I} \vdash \{Q[x/t]\}x := t\{Q\}$, suppose that $(s, o) \in Q[x/t]$. We know that all runs in $G(x := t, s, \mathcal{I})$ are finite. Since no choices need to be made in $G(x := t, s, \mathcal{I})$, the one existing strategy profile σ is trivially an equilibrium in $G(x := t, s, \mathcal{I}, \widehat{o})$ for any outcome function $\widehat{o} \in \widehat{O}_Q$, and in particular for the outcome function $\widehat{o}$ which assigns o to σ. Note that since $(s, o) \in Q[x/t]$, $(s^x_t, o) \in Q$, and hence $\widehat{o} \in \widehat{O}_Q$.

Turning to the inference rules, note that the case of composition is treated in lemma 1, and the logical consequence rule is an easy consequence of the semantic definition of $\mathcal{I} \models \{P\}\gamma\{Q\}$. For conditional branching, the conclusion follows directly from the two premises, given that $G(\texttt{if}\, B\, \texttt{then}\, \gamma_1\, \texttt{else}\, \gamma_2, s, \mathcal{I})$ is either $G_1(\gamma_1, s, \mathcal{I})$ or $G_2(\gamma_2, s, \mathcal{I})$. Finally, iteration is take care of by lemma 2. □

Note that the soundness result also holds for Nash equilibria: If in the definition of $\mathcal{I} \models \{P\}\gamma\{Q\}$ we replace SPE by NE, the above soundness result can still be proved. This is as it should be, since every subgame perfect equilibrium is also a Nash equilibrium.

5.3 Completeness

Like in the completeness proof for the standard Hoare calculus, the notion of a weakest precondition plays an important role for our calculus as well. Given interpretation $\mathcal{I}$, mechanism γ, and e-predicate Q, we define *wpre* as follows:

$$\begin{aligned} wpre(\gamma, Q, \mathcal{I}) \quad = \quad & \{(s, o) \in S_{\mathcal{I}} \times D_{\mathcal{I}} \mid \exists \widehat{o} \in \widehat{O}_Q \exists \sigma : \sigma \text{ is an SPE in} \\ & G(\gamma, s, \mathcal{I}, \widehat{o}) \text{ and } \widehat{o}(\sigma) = o\} \end{aligned}$$

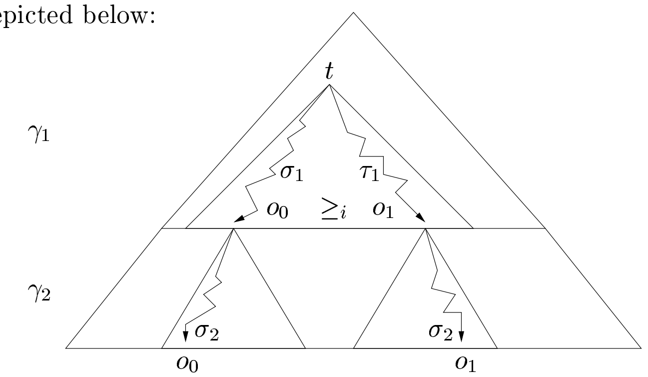
By definition, $\mathcal{I} \models \{wpre(\gamma, Q, \mathcal{I})\}\gamma\{Q\}$, and for every predicate P such that $\mathcal{I} \models \{P\}\gamma\{Q\}$, we have $P \subseteq wpre(\gamma, Q, \mathcal{I})$.

Lemma 4 (Decomposition) *If $\mathcal{I} \models \{P\}\gamma_1; \gamma_2\{Q\}$, then for some R we have $\mathcal{I} \models \{P\}\gamma_1\{R\}$ and $\mathcal{I} \models \{R\}\gamma_2\{Q\}$.*

Proof. Our assumption is $\mathcal{I} \models \{P\}\gamma_1; \gamma_2\{Q\}$. Let $R = wpre(\gamma_2, Q, \mathcal{I})$, then all we need to show is that $\mathcal{I} \models \{P\}\gamma_1\{R\}$. So supposing that $(s, o) \in P$, we need to provide an outcome function $\widehat{o}_1 \in \widehat{O}_R$ and a strategy profile σ_1 such that σ_1 is an SPE in $G_1(\gamma_1, s, \mathcal{I}, \widehat{o}_1)$ and $\widehat{o}_1(\sigma_1) = o$.

Consider the outcome function $\widehat{o} \in \widehat{O}_Q$ and the strategy profile σ for $G(\gamma_1; \gamma_2, s, \mathcal{I})$ provided by our assumption. We let $\sigma_1 = \sigma | G_1$. As for the definition of $\widehat{o}_1$, for every infinite run τ of G_1 we let $\widehat{o}_1(\tau) = \widehat{o}(\tau)$. If on the other hand τ is finite, we define $\widehat{o}_1(\tau) = \widehat{o}(\tau \cdot \sigma_\tau)$, where $\sigma_\tau = \sigma | G_\tau$. By our assumption, we have $\widehat{o}_1(\sigma_1) = \widehat{o}(\sigma) = o$. Furthermore, since $(s_\tau, \widehat{o}(\tau \cdot \sigma_\tau)) \in R$, $\widehat{o}_1 \in \widehat{O}_R$.

Hence, all we need to show is that σ_1 is an SPE in $G_1(\gamma_1, s, \mathcal{I}, \widehat{o}_1)$. So consider any subgame $G_1' = (\pi, t, \mathcal{I}, \widehat{o}_1)$ of G_1, and a strategy profile $\tau_1 \sim_i \sigma_1$, where we take $\widehat{o}_1(\sigma_1) = o_0$ and $\widehat{o}_1(\tau_1) = o_1$. Assume first that both σ_1 and τ_1 are finite. Considering $G' = (\pi; \gamma_2, t, \mathcal{I}, \widehat{o})$, we know that there is a profile σ_2 (derived from σ) such that $\sigma_1 \cdot \sigma_2$ is an SPE in G' and $\sigma_1 \cdot \sigma_2 \sim_i \tau_1 \cdot \sigma_2$. The situation is depicted below:



Also by definition, we know that $\widehat{o}_1(\sigma_1) = \widehat{o}(\sigma_1 \cdot \sigma_2) = o_0$ and $\widehat{o}_1(\tau_1) = \widehat{o}(\tau_1 \cdot \sigma_2) = o_1$, and hence we must have $o_0 \geq_i o_1$.

Note that in case either σ_1 or τ_1 or both are infinite, a simplified version of the above argument can be applied. □

The above lemma is what distinguishes subgame perfect equilibria from Nash equilibria, since only the former can be decomposed in the way shown by the decomposition lemma. For Nash equilibria, the above lemma fails: when defining $\widehat{o}_1$ in the above proof, we cannot be sure that $\widehat{o}_1 \in \widehat{O}_R$, since a subprofile of an equilibrium profile may itself not be an equilibrium profile. Consequently, also the following completeness result does not hold for Nash equilibria.

Theorem 5 (Completeness) *If* $\mathcal{I} \models \{P\}\gamma\{Q\}$ *then* $\mathcal{I} \vdash \{P\}\gamma\{Q\}$.

Proof. The proof proceeds by induction on the structure of γ. For $x := t$, note that for any state s, the game $G(x := t, s, \mathcal{I})$ contains only a single finite run ending in state s_t^x. Observe that $P \subseteq Q[x/t]$: if $(s, o) \in P$, every run terminates in state $(s_t^x, o) \in Q$, and hence $(s, o) \in Q[x/t]$. Applying the logical consequence rule to the assignment axiom, we then obtain $\mathcal{I} \vdash \{P\}x := t\{Q\}$.

For $\texttt{ch}_A$, we use the axiom and the logical consequence rule, and for $\gamma_1; \gamma_2$, we can appeal to the decomposition lemma, induction hypothesis, and the composition rule. The case of $\texttt{if}\, B\, \texttt{then}\, \gamma_1\, \texttt{else}\, \gamma_2$ is straight-forward, so we only need to deal with the while-loop.

For iteration, suppose that $\mathcal{I} \models \{P\}\texttt{while}\, B\, \texttt{do}\, \gamma\{Q\}$. Similarly, to the proof of the decomposition lemma, we let $R = \mathit{wpre}(\texttt{while}\, B\, \texttt{do}\, \gamma, Q, \mathcal{I})$. First, we shall establish that $\mathcal{I} \models \{R \cap B^{\mathcal{I}}\}\gamma\{R\}$. By definition, we have $\mathcal{I} \models \{R\}\texttt{while}\, B\, \texttt{do}\, \gamma\{Q\}$. From this, $\mathcal{I} \models \{R \cap B^{\mathcal{I}}\}\gamma; \texttt{while}\, B\, \texttt{do}\, \gamma\{Q\}$ is easily seen to follow. Now we can apply the decomposition lemma: Since the R provided by the proof of the decomposition lemma is precisely the one we defined above, we can conclude that $\mathcal{I} \models \{R \cap B^{\mathcal{I}}\}\gamma\{R\}$.

Now using the induction hypothesis and applying the while-rule, we obtain

$$\mathcal{I} \vdash \{R\}\texttt{while}\, B\, \texttt{do}\, \gamma\{R \cap \overline{B^{\mathcal{I}}}\}.$$

Since $P \subseteq R$ and $R \cap \overline{B^{\mathcal{I}}} \subseteq Q$, we can apply the logical consequence rule to derive $\mathcal{I} \vdash \{P\}\texttt{while}\, B\, \texttt{do}\, \gamma\{Q\}$. □

6 Conclusions

Two main directions for future research present themselves: On the foundational side, the question arises whether the present approach can also be applied to other equilibrium notions. We have already remarked that while the calculus presented can also be used to reason about Nash equilibria, the non-compositional nature of these equilibria stands in the way of a complete calculus.

Hence, alternative equilibrium notions that promise to be amenable to our approach will be refinements of subgame perfect equilibria. Second, we mentioned already that an intensional approach to pre- and postconditions is worth developing. For this, the crucial question is whether the logic used (FOL) and the expressiveness results obtained for programs can be carried over to mechanisms.

At the most general level, we hope that this paper has shown that tools from computational logic can be extended from program verification to the verification of game-theoretic mechanisms. The examples provided should suffice to convince the reader of the variety of possible applications of such an extension. The semantics of the correctness assertions for mechanisms is more complex than for programs, but this is counterbalanced by the fact that the mechanisms we would like to verify (e.g., spectrum auctions for telecommunication markets) may turn out to be simpler than their counterparts in computer software (e.g., operating systems).

References

[1] K.R. Apt and E.R. Olderog. *Verification of Sequential and Concurrent Programs.* Springer, second edition, 1997.

[2] S. Brams and P.C. Fishburn. Voting procedures. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare.* Elsevier, forthcoming.

[3] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[4] E. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[5] E. Dijkstra. *A Discipline of Programming.* Prentice-Hall, 1976.

[6] N. Francez. *Program Verification.* Addison-Wesley, 1992.

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

[8] S. Kraus. *Strategic Negotiation in Multiagent Environments.* MIT Press, 2001.

[9] J. Moore. Implementation, contracts, and renegotiation in environments with complete information. In J.-J. Laffont, editor, *Advances in Economic Theory: Sixth World Congress*, volume 1. Cambridge University Press, 1992.

[10] H. R. Nielson and F. Nielson. *Semantics with Applications.* Wiley, 1992.

[11] M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[12] R. Parikh. Social software. *Synthese*. To appear.

[13] R. Parikh. The logic of games and its applications. In M. Karpinski and J. van Leeuwen, editors, *Topics in the Theory of Computation*, Annals of Discrete Mathematics 24. Elsevier, 1985.

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

[15] J.S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers.* MIT Press, 1994.