Notre Dame Journal of Formal Logic
Volume 41, Number 2, 2000

THE PROPOSITIONAL LOGIC OF ELEMENTARY TASKS

GIORGI JAPARIDZE

Abstract The paper introduces a semantics for the language of propositional additive-multiplicative linear logic. It understands formulas as *tasks* that are to be accomplished by an agent (machine, robot) working as a slave for its master (user, environment). This semantics can claim to be a formalization of the resource philosophy associated with linear logic when resources are understood as agents accomplishing tasks. I axiomatically define a decidable logic **TSKp** and prove its soundness and completeness with respect to the task semantics in the following intuitive sense: **TSKp** $\vdash \alpha$ iff α can be accomplished by an agent who has nothing but its intelligence (that is, no physical resources or external sources of information) for accomplishing tasks.

1. Introduction

What we call *atomic tasks* are expressed in natural languages with imperative sentences, such as 'Open the door!' or 'Solve the P = NP problem!'. These kinds of expressions may have one of the two semantical values: *accomplished* or *failed*. These values usually are not initially known and only become fully determined in what can be called the *eventual situation*: the task 'Open the door' will be considered accomplished if the door was eventually opened, no matter when and by whom.

The connectives of classical logic can be applied to tasks with their standard behavior: the task $\alpha \to \beta$ is accomplished if and only if β is eventually accomplished as long as α is so, $\bot$ is a task that is never accomplished, and so on, so that if we only consider what we call *elementary tasks*—Boolean combinations of atomic tasks—the set of valid (always accomplished) tasks coincides with the set of classical tautologies.

Nonelementary tasks are produced by the nonclassical operator $\sqcap$. Explaining its semantics requires a little analysis of our intuitive notion of a task. When we talk about tasks, we have two agents in mind: the agent who must accomplish the task

Received May 18, 2001; printed August 30, 2002
2001 Mathematics Subject Classification: Primary, 03B47; Secondary, 03B70, 68T27
Keywords: Tasks, game semantics, linear logic, substructural logics

(open the door) and the agent who sets the task (requests 'Open the door!'). The former can be called *slave* and the latter *master*. With potential applications in mind, slave can be thought of as a machine or a robot and master as a user or the environment.

Atomic formulas are understood as requests automatically made by master. As for

$$\alpha \sqcap \beta,$$

this is a task that signifies no particular request for slave but rather two potential requests that master can make: α and β. Accomplishing the task $\alpha \sqcap \beta$ means accomplishing the task chosen by master. If master did not make any choice, then this task is considered accomplished as there was no particular request that slave failed to carry out. As for

$$\alpha \wedge \beta,$$

accomplishing it means accomplishing both α and β. This task is generally harder to carry out than $\alpha \sqcap \beta$ because the former obligates slave to carry out both of the two tasks while the latter can signify only one (even if an arbitrary one) of the two tasks. For example, the task

$$\textit{Make it faster!} \sqcap \textit{Make it slower!}$$

can be accomplished while

$$\textit{Make it faster!} \wedge \textit{Make it slower!}$$

is an unaccomplishable task.

Revisiting the operator $\rightarrow$ for the cases when its scope is no longer restricted to elementary tasks, the intuitive meaning of the task

$$\alpha \rightarrow \beta$$

is that slave should carry out β as long as master carries out α under slave's command. In other words, in α the roles of master and slave are interchanged, so that if, say, $\alpha = \alpha_1 \sqcap \alpha_2$, it is slave rather than master who makes one of the requests α_1 or α_2. For example, while being unable to accomplish the task

$$\textit{Kill the werewolf!} \sqcap \textit{Destroy the vampire!},$$

slave may have sufficient resources (such as a gun, a mallet, and bravery) to carry out the conditional task

$$\textit{(Give me a silver bullet!} \sqcap \textit{Give me a wooden stake!)} \rightarrow \textit{(Kill the werewolf!} \sqcap \textit{Destroy the vampire!).}$$

In the process of what we call a *realization* of this task, master can, at any time, request either killing the werewolf or destroying the vampire. And also at any time, slave can request either giving him a silver bullet or giving him a wooden stake. The task will be accomplished, if master's request (if there was one) was eventually satisfied, or if slave's (counter)request was not satisfied.

Generally, accomplishability of $\alpha \rightarrow \beta$ corresponds to our intuition of *reducibility* of the task β to the task α, so that $\rightarrow$ can be viewed as the task reduction operator.

This interpretation makes every task of the form $\alpha \rightarrow \alpha$ accomplishable by a sufficiently intelligent slave who does not need to have any extra physical or informational resources. We can call these sorts of tasks *universally accomplishable*. Slave's successful strategy for, say, the task

$$\textit{(Make it faster!} \sqcap \textit{Make it slower!)} \rightarrow \textit{(Make it faster!} \sqcap \textit{Make it slower!)}$$

would be to wait until master makes one of the two possible requests in the consequent and then repeat the same request in the antecedent.

Not all the tasks that have the form of a classical tautology are universally accomplishable though. For example, slave cannot have a successful strategy for

(Make it faster! $\sqcap$ *Make it slower!)* $\rightarrow$
(Make it faster! $\sqcap$ *Make it slower!)* $\wedge$ *(Make it faster!* $\sqcap$ *Make it slower!)*.

In particular, slave will fail to carry out this task if master selects different $\sqcap$-terms in the two conjuncts of the consequent.

This reminds us of linear logic and its variants that, too, reject the principle

$$\alpha \rightarrow (\alpha \wedge \alpha).$$

Our approach can serve as a semantical justification for substructural logics. In particular, it can be considered a formalization of the resource philosophy originally associated with linear logic (Girard [2]) where resources are understood as tasks that are being carried out *for* the agent (rather than *by* the agent). That is, resources are symmetric to tasks: what is a task for slave is a resource for master, and vice versa. By their "behavior," the classical operators of our language are similar to the multiplicative operators of linear logic whereas $\sqcap$ and its dual $\sqcup$ correspond to the additive conjunction and disjunction, respectively.

Even though our semantics is formulated without using game-semantical terms, from the technical point of view it can be classified as a (Blass [1]- and Japaridze [3]-style) game semantics.

What follows contains strict definitions for the task semantics at the propositional level and an axiomatization (with a soundness and completeness proof) of the corresponding logic—the set of universally accomplishable tasks.

2. Main Definitions

The language we consider is that of classical propositional logic augmented with the binary operator $\sqcap$. The connective $\sqcup$ can be defined by $\alpha \sqcup \beta = \neg(\neg\alpha \sqcap \neg\beta)$. For the considerations of compactness of definitions and proofs, we choose a minimal set of basic classical operators: $\rightarrow$ and $\bot$.

Formulas of our language we sometimes also call *tasks* and use lowercase Greek letters exclusively as metavariables for them. A formula (task) not containing $\sqcap$ is said to be *elementary*. By a $\sqcap$-*formula* we mean a formula of the form $\alpha \sqcap \beta$. The $\sqcap$-*complexity* of a formula is the number of occurrences of $\sqcap$ in that formula. A *surface occurrence* of a subformula in a formula is an occurrence that is not in the scope of $\sqcap$.

We assume some standard fixed way of referring to surface occurrences of subformulas such as, for example, expressions of the type '*Antecedent of Consequent of Antecedent*'. These sorts of expressions will be called *occurrence specifications* and uppercase Greek letters will be used as metavariables for them. What we mean by *validity* of an occurrence specification for a formula must be clear: say, for the above specification to be valid for α, α needs to have the form $(\beta \rightarrow (\delta \rightarrow \psi)) \rightarrow \gamma$ in which case the specification specifies the occurrence of δ.

If Γ is a valid occurrence specification for α, then

$$\Gamma(\alpha)$$

denotes the occurrence in α specified by Γ, to which we can also refer as 'the occurrence Γ in α'.

An occurrence specification is said to be *positive* if it specifies an occurrence that is in the antecedent of an even (possibly zero) number of $\rightarrow$'s. Otherwise it is *negative*.

If Γ is a valid occurrence specification for α and γ is any formula, we call the pair

$$\Gamma/\gamma$$

a *replacement* for α.

If $E = \Gamma/\gamma$ is a replacement for α, we can use the expression

$$\alpha(E)$$

or

$$\alpha(\Gamma/\gamma)$$

to denote the result of replacing in α the occurrence Γ by γ.

Definition 2.1

1. A *simple request* for α is a replacement Γ/γ for α such that Γ is positive, $\Gamma(\alpha) = \beta_0 \sqcap \beta_1$, and $\gamma = \beta_0$ or $\gamma = \beta_1$.
2. A *simple response* for α is defined in the same way only Γ here should be negative rather than positive.

Note that the part of the task that gets changed by a simple request or response is a surface occurrence because, as we agreed, 'occurrence specification' (Γ) always means the specification of a surface occurrence.

Intuitively, a simple request means an elementary request made by master, and a simple response means an elementary (counter)request made by slave.

Definition 2.2 A *request* for α is a sequence

$$X = \langle E_1, \ldots, E_n \rangle$$

such that E_1 is a simple request for $\alpha_0 = \alpha$, E_2 is a simple request for $\alpha_1 = \alpha_0(E_1)$, E_3 is a simple request for $\alpha_2 = \alpha_1(E_2), \ldots$.

A *response* for α is defined in the same way, only $E_1, \ldots, E_n$ here should be simple responses rather than requests.

When $X = \langle E_1, \ldots, E_n \rangle$ is a request or a response for α and $\alpha_0 = \alpha, \alpha_1 = \alpha_0(E_1)$, $\alpha_2 = \alpha_1(E_2), \ldots, \alpha_n = \alpha_{n-1}(E_n)$, we will use the expression

$$\alpha(X)$$

to denote α_n, that is, the result of consecutively making the replacements $E_1, \ldots, E_n$ in α. If here X is the empty request or response $\langle\,\rangle$, then we define $\alpha(X) = \alpha$.

By abuse of terminology, we will also say that a formula β is a *request* for α, if $\beta = \alpha(X)$ for some request X for α; we say that such a request is *proper*, if $X \neq \langle\,\rangle$. Similarly, we will say that a formula β is a *simple request* for α, if $\beta = \alpha(X)$ for some simple request X for α. The same terminological convention applies to responses and simple responses.

Thus, a request (respectively, response) for α is the result of consecutively replacing in α some (possibly none) positive (respectively, negative) surface occurrences of the form $\beta \sqcap \gamma$ by β or γ; in a simple request or response, exactly one replacement is made.

The following fact can be easily observed.

Fact 2.3 Assume X is a response and Y is a request for α. Then X is also a response for $\alpha(Y)$ and Y is also a request for $\alpha(X)$; moreover,

$$\big(\alpha(X)\big)(Y) = \big(\alpha(Y)\big)(X).$$

When X is a response (or simple response) and Y is a request (or simple request) for α, we will usually use the notation

$$\alpha(X, Y)$$

to mean the same as the less symmetric-looking notation $\big(\alpha(X)\big)(Y)$.

When $\beta = \alpha(X, Y)$ for some response X and request Y for α, we say that β is a *development* of α; if here $\beta \neq \alpha$, that is, $X \neq \langle\,\rangle$ or $Y \neq \langle\,\rangle$, then β is said to be a *proper development* of α.

Definition 2.4 A *realization* of α is a sequence

$$\langle \alpha_0, \ldots, \alpha_m \rangle$$

such that $\alpha_0 = \alpha$ and for every i with $0 \leq i < m$, α_{i+1} is a proper development of α_i.

When we simply say 'a realization', we mean a realization of α for some arbitrary formula α.

Definition 2.5 A *response strategy* is a function f that assigns, to every realization, a response for the last formula of the realization.

Thus, a response strategy is a procedure that looks at the current state of the task, together with the history of the task, and decides what response to make for it.

Definition 2.6 Let f be a response strategy. A *realization of α with f* is a realization

$$R = \langle \alpha_0, \ldots, \alpha_m \rangle$$

of α such that $f(R) = \langle\,\rangle$ and for every i with $0 \leq i < m$, we have $\alpha_{i+1} = \alpha_i(X, Y)$, where $X = f\langle \alpha_0, \ldots, \alpha_i \rangle$ and Y is an arbitrary request for α_i.

Intuitively, this is how a realization of α_0 with response strategy f is produced: f (slave) reads the current input $I = \langle \alpha_0, \ldots, \alpha_i \rangle$ (initially $i = 0$) which is the history of the task, and starts thinking what response (series of simple responses) to make for its last formula; while f is thinking, master can make zero, one, or more requests for α_i, the combination of which is still called a (one) request. Once f has come to a decision, the response it finds is combined with master's request(s) and applied to α_i getting a development β of α_i. This updates the input I to I', where $I' = \langle \alpha_0, \ldots, \alpha_i, \alpha_{i+1} \rangle$ with $\alpha_{i+1} = \beta$, except when $\beta = \alpha_i$, in which case $I' = I$. In either case, the whole procedure will now be applied to I' as a new input, and so on.

Thus, our definition allows multiple (nonsimple) requests and responses at every step, as well as simultaneous requests and responses. The only motivation for adopting this relaxed protocol is to get a symmetry between master and slave. An alternative approach would be to alternate exclusive accesses to the input between master and slave, with or without requiring requests and responses to be simple. Each of these three variants, as well as many other reasonable modifications of the protocol, would produce the same class of valid (universally accomplishable) tasks. This sort of robustness is a positive sign and it indicates how natural the semantics is, reminiscent of the situation with different definitions of Turing machines and other models of

computation that all lead to the same class of computable functions—the phenomenon that serves as a major argument in favor of the Church-Turing thesis.

Definition 2.7 An *eventual situation* is a function s that assigns to every propositional atom one of the values $\{1, 0\}$. 1 corresponds to the intuitive value *accomplished* and 0 corresponds to the value *failed*.

This function is extended to all formulas as follows:

1. $s(\bot) = 0$.
2. $s(\alpha \to \beta) = \begin{cases} 0 & \text{if } s(\alpha) = 1 \text{ and } s(\beta) = 0; \\ 1 & \text{otherwise.} \end{cases}$
3. $s(\alpha \sqcap \beta) = 1$.

The *elementarization* of α, denoted by

$$\overline{\alpha},$$

is the result of replacing in α all (surface) occurrences of all $\sqcap$-subformulas by $\top$ which abbreviates $\bot \to \bot$. The following fact immediately follows from Definition 2.7.

Fact 2.8 For every eventual situation s, we have $s(\alpha) = s(\overline{\alpha})$.

An eventual situation s can be thought of as a classical model where an atom p is interpreted as a true proposition if and only if $s(p) = 1$.

When restricted to elementary formulas, Definition 2.7 is the same as the classical definition of truth in model s. Therefore, for every elementary formula α, we have $s(\alpha) = 1$ if and only if α is true in s in the classical sense. Thus we have the following.

Fact 2.9 An elementary formula α is a classical tautology if and only if $s(\alpha) = 1$ for every eventual situation s.

Observe that development preserves the $\to$-structure of the formula. That is, a development of $\alpha \to \beta$ always has the form $\alpha' \to \beta'$ where α' and β' are developments of α and β, respectively. It follows that if an occurrence specification is valid for a formula, it is also valid for any development of that formula.

Assume

$$R = \langle \alpha_0, \ldots, \alpha_m \rangle$$

is a realization of α_0 and Γ is a valid occurrence specification for α_0. Then the *projection of R on Γ* denoted by

$$\Gamma(R),$$

is the result of deleting in

$$\langle \Gamma(\alpha_0), \ldots, \Gamma(\alpha_m) \rangle$$

every formula that is a duplicate of its predecessor in the sequence. It is obvious that $\Gamma(R)$ is a realization of $\Gamma(\alpha_0)$.

Definition 2.10 We say that a realization

$$R = \langle \alpha_0, \ldots, \alpha_m \rangle$$

of a formula α_0 is *successful* with respect to an eventual situation s, if one of the following conditions holds:

1. α_0 is atomic (which implies $m = 0$) and $s(\alpha_0) = 1$;
2. $\alpha_0 = \beta \to \gamma$ and *Consequent*(R) is successful with respect to s whenever *Antecedent*(R) is so;

3. α_0 is a $\sqcap$-formula and either $m = 0$ or $m \geq 1$ and $\langle \alpha_1, \ldots, \alpha_m \rangle$ is successful with respect to s.

This definition formalizes the intuition on accomplishing a task that was described in Section 1. In the context of an actual realization $\langle \alpha_0, \ldots, \alpha_m \rangle$ and an actual eventual situation s, the task α_0 should be considered accomplished if the realization is successful with respect to the eventual situation. In particular, an atomic task is accomplished if and only if it has the value 1 in s. The task $\beta \rightarrow \gamma$ is accomplished if and only if γ (the projection of the realization on the consequent) is accomplished as long as β (the projection on the antecedent) is accomplished. The task α_0 of the form $\beta \sqcap \gamma$ is accomplished if and only if either there was no command specifying which particular subtask implied by α_0 should be accomplished or there was such a command and the specified subtask was accomplished.

Lemma 2.11 *A realization of a task is successful with respect to an eventual situation s if and only if s assigns the value* 1 *to the last formula of the realization.*

Proof: Assume R is a realization of a task α and s is an eventual situation. Below 'successful' means 'successful with respect to s'.

We use induction on the complexity of α. If α is atomic, then by Definition 2.10, R is successful if and only if $s(\alpha) = 1$. It remains to notice that $R = \langle \alpha \rangle$ so that α is (the first and) the last formula of R.

Assume $\alpha = \beta_0 \rightarrow \gamma_0$. Then R has the form

$$\langle \beta_0 \rightarrow \gamma_0, \ldots, \beta_m \rightarrow \gamma_m \rangle.$$

Therefore β_m is the last formula of *Antecedent*(R) as the latter is nothing but $\langle \beta_0, \ldots, \beta_m \rangle$ with duplicate formulas removed. Similarly, γ_m is the last formula of *Consequent*(R). *Antecedent*(R) and *Consequent*(R) are realizations of β_0 and γ_0, respectively. The complexities of β_0 and γ_0 are lower than that of α. Hence, by the induction hypothesis, *Antecedent*(R) is successful if and only if $s(\beta_m) = 1$, and *Consequent*(R) is successful if and only if $s(\gamma_m) = 1$. Therefore, in view of clause 2 of Definition 2.7, $s(\beta_m \rightarrow \gamma_m) = 1$ if and only if *Antecedent*(R) is not successful or *Consequent*(R) is successful. But by Definition 2.10, this is the case if and only if R is successful. Thus, R is successful if and only if s assigns 1 to its last formula $\beta_m \rightarrow \gamma_m$.

Finally, assume α is $\beta \sqcap \gamma$. If $R = \langle \alpha \rangle$, then by Definition 2.10, R is successful and by Definition 2.7, $s(\alpha) = 1$. If $R = \langle \alpha, \alpha_1, \ldots, \alpha_m \rangle$ ($m \geq 1$), then α_1 must be a proper development of α and hence have a lower complexity than α does. Therefore, by the induction hypothesis, the realization $\langle \alpha_1, \ldots, \alpha_m \rangle$ of α_1 is successful if and only if $s(\alpha_m) = 1$. By Definition 2.10, R is successful if and only if $\langle \alpha_1, \ldots, \alpha_m \rangle$ is so. Thus, R is successful if and only if $s(\alpha_m) = 1$. □

The reader may ask why I have not chosen Lemma 2.11 as a definition of 'successful' instead of Definition 2.10. There are two strong reasons: First of all, it is Definition 2.10 rather than Lemma 2.11 that captures the task intuition described in Section 1. Secondly, the robustness/portability of the definition would be lost. The point is that Definition 2.10 has natural generalizations for certain more expressive languages—languages where realizations may no longer be finite which would make Lemma 2.11 inapplicable.

Definition 2.12 We say that a response strategy f *universally accomplishes* α if and only if for every eventual situation s, every realization of α with f is successful with respect to s. If such a response strategy f exists, we say that α is *universally accomplishable*.

Definition 2.13 (Logic TSKp) The *axioms* are all the elementary formulas that are classical tautologies. The *rules of inference* are:

RS-rule: $\dfrac{\pi}{\alpha}$,

where π is a simple response for α;

RQ-rule: $\dfrac{\overline{\alpha},\pi_1,\ldots,\pi_n}{\alpha}$,

where ($\overline{\alpha}$ is the elementarization of α and) $\pi_1, \ldots, \pi_e$ are all the simple requests for α.

By a straightforward induction on the $\sqcap$-complexity of a formula the following can be verified.

Fact 2.14 **TSKp** is decidable.

3. Main Theorem

Theorem 3.1 *A task is provable in* **TSKp** *if and only if it is universally accomplishable.*

The rest of this section is devoted to a proof of this theorem. In this proof we will be using the expression

$$\alpha(\beta)$$

to denote a formula α together with a certain surface occurrence of a subformula β. This notation fixes that particular occurrence so that using, in the same context, the expression $\alpha(\gamma)$ would mean the result of replacing that particular occurrence of β by γ in α.

In the same style,

$$\alpha(\beta_1, \ldots, \beta_n)$$

will denote α together with certain occurrences of subformulas $\beta_1, \ldots, \beta_n$. Note that for some $i \neq j$, we may have $\beta_i = \beta_j$, but still β_i and β_j are different occurrences that may get replaced with different γs in $\alpha(\gamma_1, \ldots, \gamma_n)$. It should be seen from the context whether, when referring to β_i, we mean β_i as a formula or as a particular occurrence of that formula. The expression

$$\mathbf{TSKp} \vdash_l \alpha$$

will be used to say that α has a proof in **TSKp** whose length is at most l.

Lemma 3.2 *If* $\mathbf{TSKp} \vdash_l \alpha$ *and* ρ *is a simple request for* α*, then* $\mathbf{TSKp} \vdash_{l-1} \rho$.

Proof: We proceed by induction on l. If $l = 1$, that is, α is an axiom, then α has no simple requests and we are done. Assume now $l > 1$ and ρ is a simple request for α.

If α is obtained by the RQ-rule from $\overline{\alpha}, \pi_1, \ldots, \pi_n$, then $\rho = \pi_i$ for one of the i with $1 \leq i \leq n$. Each of these π_i has a proof of length $\leq (l-1)$ and hence, so does ρ.

Suppose now α is obtained by the RS-rule from π. Since ρ is a simple request for α and π is a simple response for α, we must have

$$\alpha = \alpha(\xi_0 \sqcap \xi_1,\ \eta_0 \sqcap \eta_1),$$

$$\rho = \alpha(\xi_0 \sqcap \xi_1,\ \eta_j),$$

and

$$\pi = \alpha(\xi_i,\ \eta_0 \sqcap \eta_1)$$

for some $\xi_0, \xi_1, \eta_0, \eta_1$ and $i,\ j \in \{0, 1\}$ where $\xi_0 \sqcap \xi_1$ is negative and $\eta_0 \sqcap \eta_1$ is positive in α. Let then

$$\theta = \alpha[\xi_i,\ \eta_j].$$

Obviously θ is a simple request for π and a simple response for ρ. That θ is a simple request for π implies, by the induction hypothesis, that θ has a shorter proof than π does so that, since $\mathbf{TSKp} \vdash_{l-1} \pi$, we have $\mathbf{TSKp} \vdash_{l-2} \theta$. Consequently, since ρ can be derived from θ by the RS-rule, we have $\mathbf{TSKp} \vdash_{l-1} \rho$. □

To show the soundness of **TSKp**, assume $\mathbf{TSKp} \vdash_l \alpha$. Below we define a response strategy f and show, by induction on l, that it universally accomplishes α.

Case 1 α is an axiom, that is, $l = 1$, that is, α is a classical tautology. Let then f be an arbitrary strategy, say, the one that always returns $\langle\ \rangle$.

As α is elementary, it has no proper developments and $\langle\alpha\rangle$ is the only possible realization of α with f. Therefore, in view of Lemma 2.11, $\langle\alpha\rangle$ is successful with respect to an eventual situation s if and only if $s(\alpha) = 1$. But since α is a tautology, by Fact 2.9, for every eventual situation s, we have $s(\alpha) = 1$. Thus, f universally accomplishes α.

Case 2 $\alpha = \alpha(\xi_0 \sqcap \xi_1)$ follows from $\pi = \alpha(\xi_i)$ by the RS-rule. The proof of π is shorter than l and, by Lemma 3.2, every request for π also has a proof shorter than l. Let

$$\beta_1, \ldots, \beta_k$$

be all the distinct requests for π. By the induction hypothesis, there are response strategies

$$f_1, \ldots,\ f_k$$

that universally accomplish $\beta_1, \ldots, \beta_k$, respectively.

Let then f be a function that acts as follows:

1. $f\langle\alpha\rangle = \langle\xi_0 \sqcap \xi_1 \,/\, \xi_i\rangle$;
2. For any realization $\langle\alpha, \alpha_1, \ldots, \alpha_m\rangle$ with $m \geq 1$ where α_1 is one of the β_j $(1 \leq j \leq k)$, f returns the same value as f_j does for the realization $\langle\alpha_1, \ldots, \alpha_m\rangle$.

We claim that f universally accomplishes α. Indeed, observe that $\alpha(f\langle\alpha\rangle) = \pi$. Therefore, every realization of α with f has the form

$$\langle\alpha, \pi', \theta_1, \ldots, \theta_m\rangle$$

$(m \geq 0)$ where π' is a request for (both α and) π. Remember that $\beta_1, \ldots, \beta_k$ are all the distinct requests for π, so we must have $\pi' = \beta_j$ for one of the j, $1 \leq j \leq k$. Thus, the realization of α has the form

$$\langle\alpha, \beta_j, \theta_1, \ldots, \theta_m\rangle.$$

From the way f is defined, it is easy then to see that $\langle \beta_j, \theta_1, \ldots, \theta_m \rangle$ is a realization of β_j with the response strategy f_j. It was our assumption that the latter universally accomplishes β_j. Therefore, by Lemma 2.11, the last formula of $\langle \beta_j, \theta_1, \ldots, \theta_m \rangle$ has the value 1 for every eventual situation s. But the last formula of $\langle \beta_j, \theta_1, \ldots, \theta_m \rangle$ is the same as the last formula of $\langle \alpha, \pi', \theta_1, \ldots, \theta_m \rangle$ which implies that the latter is successful with respect to every eventual situation s. Thus, every realization of α with f is successful with respect to every eventual situation, so f universally accomplishes α.

Case 3 α follows from $\overline{\alpha}, \pi_1, \ldots, \pi_n$ by the RQ-rule. Let

$$\beta_1, \ldots, \beta_k$$

be all the distinct proper requests for α (so that $\pi_1, \ldots, \pi_n$ are among them). It follows from Lemma 3.2 that each of these formulas has a proof shorter than l and hence, by the induction hypothesis, there are response strategies

$$f_1, \ldots, f_k$$

that universally accomplish $\beta_1, \ldots, \beta_k$, respectively.

Let then f be a function that acts as follows:

1. $f\langle \alpha \rangle = \langle\,\rangle$;
2. For any realization $\langle \alpha, \alpha_1, \ldots, \alpha_m \rangle$ with $m \geq 1$ where α_1 is one of the β_j $(1 \leq j \leq k)$, f returns the same value as f_j does for the realization $\langle \alpha_1, \ldots, \alpha_m \rangle$.

We claim that f universally accomplishes α. Indeed, consider an arbitrary realization $\langle \alpha, \theta_1, \ldots, \theta_m \rangle$ of α with the response strategy f and an arbitrary eventual situation s. We need to show that $\langle \alpha, \theta_1, \ldots, \theta_m \rangle$ is successful with respect to s.

There are two subcases to consider.

Subcase 3.1 Assume $m = 0$. Then α is the last formula of the realization and by Lemma 2.11 the realization is successful with respect to s if and only if $s(\alpha) = 1$. So it suffices to show that $s(\alpha) = 1$. But indeed, since $\overline{\alpha}$ is an elementary formula provable in **TSKp**, it must be a tautology (otherwise there would be no way to derive it in **TSKp**). Hence, by Fact 2.9, $s(\overline{\alpha}) = 1$ whence, by Fact 2.8, $s(\alpha) = 1$.

Subcase 3.2 Assume now $m \geq 1$. Since $f\langle \alpha \rangle = \langle\,\rangle$, θ_1 must be a proper request for α. As $\beta_1, \ldots, \beta_k$ are all the distinct proper requests for α, we must have $\theta_1 = \beta_j$ for one of the j, $1 \leq j \leq k$. Then it can be easily seen from the way f is defined that $\langle \theta_1, \ldots, \theta_m \rangle$ is a realization of θ_1 with the response strategy f_j. It was our assumption that the latter universally accomplishes β_j and hence θ_1. Therefore, by Lemma 2.11, $s(\theta_m) = 1$. And since θ_m is also the last formula of $\langle \alpha, \theta_1, \ldots, \theta_m \rangle$, this realization is also successful with respect to s.

This completes the soundness part of our theorem. Our remaining task now is to prove the completeness part.

Lemma 3.3 *Assume f is a response strategy, $f\langle \alpha \rangle = X$, Y is a request for α and $\alpha(X, Y)$ is not universally accomplishable. Then f does not universally accomplish α.*

Proof: Assume the conditions of the lemma. It is sufficient to consider the only nontrivial case when $\alpha(X, Y) \neq \alpha$. Assume for a contradiction that f universally accomplishes α. Let f' be the response strategy such that for any realization $\langle \alpha_0, \ldots, \alpha_k \rangle$, where α_0 is a proper development of α,

$$f'\langle \alpha_0, \ldots, \alpha_k \rangle = f\langle \alpha, \alpha_0, \ldots, \alpha_k \rangle.$$

Since $\alpha(X, Y)$ is not universally accomplishable, there is a situation s and a realization

$$R = \langle \alpha(X, Y),\ \alpha_1,\ \ldots,\ \alpha_k \rangle$$

of $\alpha(X, Y)$ with strategy f' such that R is unsuccessful with respect to s. But notice that then

$$\langle \alpha, \alpha(X, Y), \alpha_1, \ldots, \alpha_k \rangle$$

is a realization of α with strategy f. By Lemma 2.11, the latter is also unsuccessful with respect to s because it has the same last formula as R does which is in contradiction with our assumption that f universally accomplishes α. □

To prove the completeness of **TSKp**, assume **TSKp** $\nvdash \alpha$. We are going to show that α is not universally accomplishable by induction on the $\sqcap$-complexity of α.

Basis: Assume α is elementary. **TSKp** $\nvdash \alpha$ implies that α is not a tautology. In view of Fact 2.9, this means that here is an eventual situation s such that $s(\alpha) = 0$. As α is elementary, the only realization of it is $\langle \alpha \rangle$ and by Lemma 2.11 this realization is not successful with respect to s. Thus, α is not universally accomplishable.

Inductive step: Assume α is not elementary. Let f be an arbitrary response strategy. We need to show that f does not universally accomplish α, that is, there is a realization of α with f that is not successful with respect to some eventual situation.

There are two cases to consider.

Case 1 $\overline{\alpha}$ is not a tautology. Then, by Facts 2.9 and 2.8, there is an eventual situation s such that

$$s(\alpha) = 0. \tag{1}$$

If $f\langle \alpha \rangle = \langle\ \rangle$, then $\langle \alpha \rangle$ is a realization of α. It follows from (1) by Lemma 2.11 that this realization is not successful with respect to s.

Assume now $f\langle \alpha \rangle = X$ for some nonempty response X for α. Then **TSKp** $\nvdash \alpha(X)$ because otherwise, by the RS-rule, we would have **TSKp** $\vdash \alpha$. Also, the $\sqcap$-complexity of $\alpha(X)$ is lower than that of α. Therefore, by the induction hypothesis, $\alpha(X)$ is not universally accomplishable. Then by Lemma 3.3 (with $Y = \langle\ \rangle$), f does not universally accomplish α.

Case 2 $\overline{\alpha}$ is a tautology. Since α is not provable in **TSKp**, there is a simple request for α that is not provable (otherwise α would be derivable by the RQ-rule). Thus we have $\alpha = \alpha(\xi_0 \sqcap \xi_1)$, where $\xi_0 \sqcap \xi_1$ is a positive occurrence, and $\alpha(\xi_i)$ is not provable for $i = 0$ or $i = 1$.

Clearly a response for $\alpha(\xi_i)$ cannot be provable, for otherwise, by the RS-rule, $\alpha(\xi_i)$ would be provable, too. Therefore, by the induction hypothesis we have

$$\text{No response for } \alpha(\xi_i) \text{ is universally accomplishable.} \tag{2}$$

Denote by Y the request $\langle \xi_0 \sqcap \xi_1 \,/\, \xi_i \rangle$ so that $\alpha(Y) = \alpha(\xi_i)$. Consider an arbitrary response strategy f. Let $f\langle\alpha\rangle = X$. In view of Fact 2.3, X would also be a response for $\alpha(Y)$. Therefore, by (2), $\alpha(X, Y)$ is not universally accomplishable. Then, by Lemma 3.3, f does not universally accomplish α.

Theorem 3.1 is proven.

4. Announcing Further Results

We call **TSKp** the logic of elementary tasks because its atoms are meant to represent only elementary tasks: **TSKp** is closed under substitution of propositional letters with elementary tasks but not nonelementary tasks. For example, where p and q are propositional letters, $\mathbf{TSKp} \vdash p \to p \wedge p$ but $\mathbf{TSKp} \nvdash (p \sqcap q) \to (p \sqcap q) \wedge (p \sqcap q)$. Therefore, the logic whose propositional letters are meant to represent any (elementary or nonelementary) tasks, and which is thus closed under unrestricted substitution for atoms, would be different from **TSKp**. Let us call that logic the *propositional logic of valid task schemata*.

Claim 4.1 *The propositional logic of valid task schemata is decidable and it is exactly the propositional fragment of logic* **ET**.

ET is a decidable first-order logic introduced by the author in [3] with a different semantics in mind. This logic is a proper extension of Affine logic (additive-multiplicative linear logic + weakening) and its language, in addition to the connectives (equivalent to those) of the language of **TSKp**, has the quantifier-type operator $\sqcap$. In the context of our task semantics, $\sqcap x\alpha(x)$ can be interpreted as $\alpha(a_0) \sqcap \alpha(a_1) \sqcap \alpha(a_2) \sqcap \cdots$, where $a_0, a_1, a_2, \ldots$ are all the objects of the universe of discourse. I have not formally introduced $\sqcap$ as the present paper is focused on propositional logic only, but extending definitions to $\sqcap$ does not present a problem. Then the propositional logic of valid task schemata naturally extends to the *predicate logic of valid task schemata*; as it turns out, so does Claim 4.1.

Claim 4.2 *The predicate logic of valid task schemata is decidable and it is exactly* **ET**.

The author intends to publish the proofs of Claims 4.1 and 4.2 in a separate paper. The (a little lengthy) definition of **ET** is not included here either and an interested reader can look it up in [3].

I would like to finish the paper with three examples of formulas that separate **ET** from Affine logic:

$$(p \wedge q) \vee (r \wedge s) \;\to\; (p \vee r) \wedge (q \vee s);$$

$$\big(p \wedge (r \sqcap s)\big) \sqcap \big(q \wedge (r \sqcap s)\big) \sqcap \big((p \sqcap q) \wedge r\big) \sqcap \big((p \sqcap q) \wedge s\big) \;\to\; (p \sqcap q) \wedge (r \sqcap s);$$

$$\sqcap x\Big(\big(P(x) \wedge \sqcap x\, Q(x)\big) \sqcap \big(Q(x) \wedge \sqcap x\, P(x)\big)\Big) \;\to\; \sqcap x\, P(x) \wedge \sqcap x\, Q(x).$$

References

[1] Blass, A., "A game semantics for linear logic," *Annals of Pure and Applied Logic*, vol. 56 (1992), pp. 183–220. Zbl 0763.03008. MR 93e:03041. 173

[2] Girard, J.-Y., "Linear logic," *Theoretical Computer Science*, vol. 50 (1987), pp. 1–102. Zbl 0625.03037. MR 89m:03057. 173

[3] Japaridze, G., "A constructive game semantics for the language of linear logic," *Annals of Pure and Applied Logic*, vol. 85 (1997), pp. 87–156. Zbl 0882.03057. MR 98j:03086. 173, 182

Acknowledgments

This paper was supported by a Summer Research Grant from Villanova University.

Department of Computing Sciences
Villanova University
800 Lancaster Avenue
Villanova PA 19085
japaridz@csc.villanova.edu
http://www.csc.vill.edu/~japaridz