University of Amsterdam
Institute of Logic Language and Computation

An Extension of Game Logic with Parallel Operators

Master's thesis of: Iouri Netchitailov

Thesis supervisors: Johan van Benthem
Marc Pauly

Amsterdam
The Netherlands
23rd of November 2000

1 Introduction

In this thesis we extend the semantics of Game Logic and introduce some axioms to allow the description of parallel games. Game Logic, introduced by Parikh (1985), is an extension of Propositional Dynamic Logic. Game Logic is used to study the general structure of arbitrary determined two-player games. The syntax of Game Logic contains the operators of *test*, *choice*, *sequential composition*, *duality*, and *iteration*. In particular, one can find examples of the semantics, syntax and axiomatics of Game Logic in works of Parikh (1985) and Pauly (1999). Reasoning about games is similar to reasoning about programs or processes behaviour, which is supported by such formalisms as Propositional Dynamic Logic, or Process Algebra. However, Game Logic does not accept all operators which are involved, for instance, in Process Algebra; in particular, Game Logic does not contain the parallel operators, such as *parallel composition*, or *left merge*. Thus, our idea is to introduce parallel operators for Game Logic. To realise this we explore two versions of parallelism represented in Process Algebra and Linear Logic.

Process Algebra was the first system that attracted our attention. It contains *alternative* and *sequential compositions* in the basic part that closely resemble respectively the operator of choice and sequential composition of Game Logic. Besides, general Process Algebra contains several sorts of parallel operators which we are looking for. However, it turns out that it is not easy to incorporate these operators into Game Logic immediately. There are several difficulties: one of them is that the semantics of Process Algebra does not care by which agent the processes execute: this demands a sophisticated technique to convert it into the semantics of a two-player game. The other is that the semantics of Process Algebra has a poor support of truth-values, that is to say, while one might connect falsum with deadlock, the operator of negation or duality has no an analogue. Still, a look at Process Algebra is useful, because the semantics of the parallel operators contains some features that do not appear directly in Linear Logic, the next formal system which we traced on the matter of parallel operators. Namely, it gives an option to distinguish between communicative and non-communicative parallel operators, and among different ways of parallel execution.

Additive Linear Logic resembles a Game Logic without the operators of test, sequential composition and iteration. In that sense it looks more removed from Game Logic than Basic Process Algebra. Multiplicative Linear Logic brings us a couple of parallel operators: *tensor* and *par*, and some operators of repetition: '*of course*' and '*why not*'. Taking into account that Blass (1992) and Abramsky, Jagadeesan (1994) introduced a two-player game semantics for Linear Logic, our problem of translating this semantics to Game Logic becomes more evident. However, even in that case there is a difficulty, which appears due to the fact that the semantics of Game Logic uses a description in terms of φ^I *-strategies*, whereas the semantics of Linear Logic refers to *winning strategies*. They could be connected with each other in such a manner that a φ^I -strategy can be considered as a winning strategy as well as a loosing strategy, of one player, of the other player or even of both of them. So the use of φ^I -strategies gives rise to a more complex structure in game semantics: nevertheless, we introduce an extension of the semantics of Game Logic by using analogues of the parallel operators of Linear Logic.

We explore the extension of Game Logic with parallel operators in the thesis in four chapters, besides the introduction (Chapter 1). In Chapter 2 we consider the models and the definitions of key operators of Process Algebra, emphasising the parallel one. Chapter 3 we devote to the semantics of game operators for Linear Logic based on the works of Blass (1992) and Abramsky, Jagadeesan (1994). In Chapter 4, the main part of the thesis, we introduce the semantics for parallel operators of Game Logic based on the analogues of tensor and par of Linear Logic. We also describe standard Game Logic operators in terms of the proposed semantics. Moreover, we propose some axioms for the parallel operators and offer proofs of soundness. In Chapter 5 the discussion and conclusion can be found.

2 Parallel processes in Process Algebra

The search for new prospects in semantics, syntax, and axiomatics for the operators of Game Logic we start by considering the operators of Process Algebra, which contains parallel ones.

2.1 Introduction

Programming used to make a division between data and code. Since Basic Process Algebra claims to describe the process behaviour, we can note that Basic Process Algebra covers only code and no data structure. The models for processes can be represented as by means of graphs as well as algebraically. Sometimes it is easier to capture the idea of a process through a graph (if the graph is not too large), while an algebraic representation is usually used for manipulations of the processes.

In this chapter we present some basic concepts of Process Algebra. First, in section 2.2 we give definitions of models and axioms of Basic Process Algebra. Then, in sections 2.3 and 2.4 we write definitions for parallel operators of Process Algebra without and with communication respectively.

2.2 Elements of Basic Process Algebra

Kernel of Basic Process Algebra describes finite, concrete, sequential non-deterministic processes. It could be extended by deadlock or recursion but we will not do it because such formalisms lead away from the basic concepts of Game Logic, which we would like to extend by analogy with existing models.

2.2.1 Syntax of terms

The elementary notion of the Basic Process Algebra is the notion of an *atomic action* that represents a behaviour, which description does not include operators of Basic Process Algebra, i.e. indivisible (has no structure) for the language of Basic Process Algebra. That is a single step such as reading or sending a data, renaming, removing etc. Let A is a set of the atomic actions. Atomic action is supposed to be internally non-contradictory, i.e. each atomic action $\boldsymbol{v}$ also can be considered as an algorithm, which can execute it, and always terminates successfully. It could be expressed either by process graph:

$$\begin{tikzcd}\boldsymbol{v} \arrow[d, "\boldsymbol{v}"'] \\ \surd\end{tikzcd}$$

Figure 2.1 Process graph for an atomic action

or by the predicate $\boldsymbol{v} \xrightarrow{\boldsymbol{v}} \surd$. Here the atomic action in the node (or on the left side of predicate) we can consider as a program code, and the atomic action to the left from the arc (above the arrow) as an execution or run of that code. We will not involve the processes that do notterminate.

The notion of *process term* is the next inductive level of the formalism of Basic Process Algebra.

Definition 2.1 ***Process term***

1) An atomic action is a process term.
2) *Alternative composition* $\boldsymbol{x + y}$ of process terms, that represents the process, which executes either process term $\boldsymbol{x}$ or process term $\boldsymbol{y}$, is also a process term.
3) *Sequential composition* $\boldsymbol{x \bullet y}$ of the process terms, that represents the process, which first executes

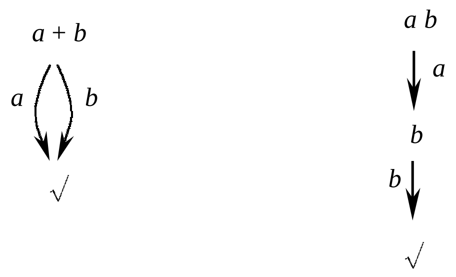


Figure 2.2 Process graphs of alternative and sequential compositions

the process term $\boldsymbol{x}$, and than the process term $\boldsymbol{y}$, is also a process term.

Each finite process can be represented by such process terms that are called *basic process terms*. The set of the basic process terms defines *Basic Process Algebra*.

2.2.2 Semantics

Intuition that underlies the definition of basic process terms can allow us to construct process graphs that correspond to these notions. By analogy with structural operational semantics (Fokkink, 2000) transition rules represented in Table 2.1 can be obtained. Here the variables $\boldsymbol{x, y, x', y'}$ are from the set of basic process terms, while $\boldsymbol{v}$ – from the set of atomic actions.

Table 2.1 Transition rules of Basic Process Algebra

$$\frac{}{v \xrightarrow{v} \surd}$$

$$\frac{x \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x'}{x + y \xrightarrow{v} x'} \qquad \frac{y \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd} \qquad \frac{y \xrightarrow{v} y'}{x + y \xrightarrow{v} y'}$$

$$\frac{x \xrightarrow{v} \surd}{x \bullet y \xrightarrow{v} y} \qquad \frac{x \xrightarrow{v} x'}{x \bullet y \xrightarrow{v} x' \bullet y}$$

The first transition rule expresses that each atomic action can terminate successfully by executing itself. The next four transition rules are for alternative composition and show that it executes either $\boldsymbol{x}$ or $\boldsymbol{y}$. The last two transition rules correspond to sequential composition and show that in that case first the process $\boldsymbol{x}$ has to be terminated successfully; only after that the process $\boldsymbol{y}$ can be started. Note that the execution of $\boldsymbol{y}$ is probably to involve another transition rule; the rules for sequential composition do not execute second term. It is a good support of the idea that the right distributive law does not satisfy to Basic Process Algebra.

2.2.3 Bisimulation Equivalence

Basic Process Algebra does not conclude that two processes are equal if they can execute exactly the same strings of actions (*trace equivalence*). As a reason we can look at the example of the graphs in Figure 2.1 that depict the law of right distribution. If we look at an alternative composition such as an example of choice function, then it becomes substantial where to make a decision: before the reading of data or after.

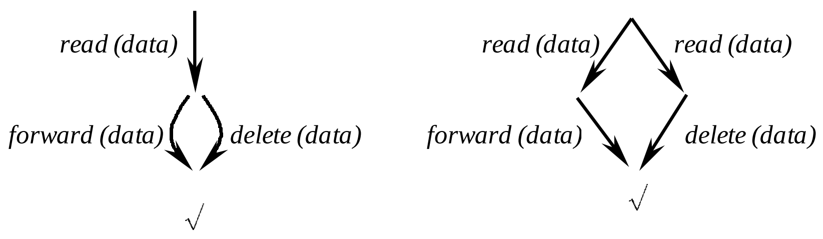


Figure 2.3 Graph's analogy of right distributive law. The two graphs are not bisimilar.

In spite of trace equivalence, bisimulation equivalence also pays attention to the branching structure of the processes.

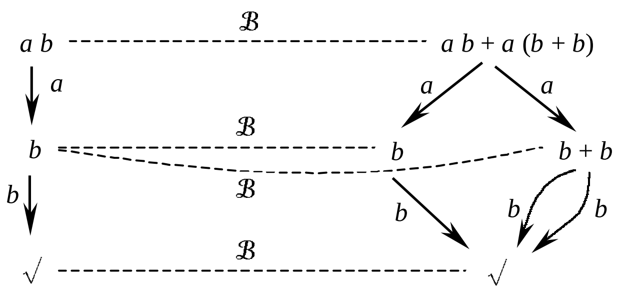


Figure 2.4 An example of bisimilar process graphs.

Definition 2.2 ***Bisimulation***

A bisimulation relation $\mathcal{B}$ is a binary relation on the processes such that:

1. if $\boldsymbol{x}\,\mathcal{B}\,\boldsymbol{y}$ and $\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \boldsymbol{x}'$, then $\boldsymbol{y} \xrightarrow{\boldsymbol{v}} \boldsymbol{y}'$ with $\boldsymbol{x}'\,\mathcal{B}\,\boldsymbol{y}'$

2. if $x \mathcal{B} y$ and $y \xrightarrow{v} y'$, then $x \xrightarrow{v} x'$ with $x' \mathcal{B} y'$

3. if $x \mathcal{B} y$ and $y \xrightarrow{v} \surd$, then $x \xrightarrow{v} \surd$

4. if $x \mathcal{B} y$ and $y \xrightarrow{v} \surd$, then $x \xrightarrow{v} \surd$

2.2.4 Axioms

Basic Process Algebra has a modulo bisimulation equivalence sound and complete *axiomatisation*, written down in Table 2.2 (Fokkink, 2000).

Table 2.2 Axioms for Basic Process Algebra

A1	$x + y$	=	$y + x$
A2	$(x + y) + z$	=	$x + (y + z)$
A3	$x + x$	=	x
A4	$(x + y) \bullet z$	=	$x \bullet z + y \bullet z$
A5	$(x \bullet y) \bullet z$	=	$x \bullet (y \bullet z)$

2.3 Operators of Process Algebra for parallel processes

Process algebra involves *parallelism* that is both non-communicative and communicative. There are several operators to express the idea of parallelism that can be treated quite widely. At first sight it is possible to find a kind of parallelism even in the *alternative composition* that is one of the operators of Basic Process Algebra and represents an action, which can choose an execution of one of the two process terms. However alternative composition can be considered as parallel only in stasis that is before choice when we have two "parallel" branches – two potentials. But during execution we will have a trace only via one branch, and all "parallelism" disappears.

2.3.1 Free merge or parallel composition

A new idea (in comparison with Basic Process Algebra) was used in the *free merge* operator or *parallel composition* introduced by Milner (1980). Free merge allows us to execute two process terms in parallel.

Table 2.3 Transition rules of Free Merge

$\dfrac{x \xrightarrow{v} \surd}{x \parallel y \xrightarrow{v} y}$	$\dfrac{x \xrightarrow{v} x'}{x \parallel y \xrightarrow{v} x' \parallel y}$
$\dfrac{y \xrightarrow{v} \surd}{x \parallel y \xrightarrow{v} x}$	$\dfrac{y \xrightarrow{v} y'}{x \parallel y \xrightarrow{v} x \parallel y'}$

The operator gives a freedom to switch between the processes during the execution. In spite of the operator of alternative composition, both terms have to be executed. In spite of the operator of sequential composition, an execution of the right process can be started before the termination of the left one. Semantics of free merge can be expressed by the transition rules represented in Table 2.3. Whereas $\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \surd$ represents a successful termination of the process term $\boldsymbol{x}$ after the execution of an atomic action $\boldsymbol{v}$, $\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \boldsymbol{x}'$ means a partial successful termination of the process term $\boldsymbol{x}$ with the remaining part $\boldsymbol{x}'$ after the execution of an atomic action $\boldsymbol{v}$. The variables $\boldsymbol{x}$, $\boldsymbol{x}'$, $\boldsymbol{y}$, and $\boldsymbol{y}'$ range over the collection of the

$$\begin{tikzcd}
 & & & & (a\,b)\parallel(b\,a) \arrow[dlll, "a"'] \arrow[drr, "b"] & & & & \\
 & b\parallel(b\,a) \arrow[dl, "b"'] \arrow[dr, "b"] & & & & & (a\,b)\parallel a \arrow[dl, "a"'] \arrow[drr, "a"] & & \\
b\,a \arrow[d, "b"'] & & b\parallel a \arrow[dl, "b"'] \arrow[dr, "a"] & & & b\parallel a \arrow[dl, "b"'] \arrow[dr, "a"] & & & a\,b \arrow[d, "a"'] \\
a \arrow[d, "a"'] & a \arrow[d, "a"'] & & b \arrow[d, "b"'] & a \arrow[d, "a"'] & & b \arrow[d, "b"'] & & b \arrow[d, "b"'] \\
\surd & \surd & & \surd & \surd & & \surd & & \surd
\end{tikzcd}$$

Figure 2.5 Process graphs of free merge (*a b*) || (*b a*)

process terms, while $\boldsymbol{v}$ ranges over the set A of the atomic actions.

Free merge can be treated as a superposition of two operators of left merge, which we will describe at the next subsection. Unfortunately, Process Algebra with free merge has not finite sound and complete axiomatisation. This fact was proved by Moller (1990).

2.3.2 Left merge

Left merge can be considered only as a decomposition of free merge. It has not an independent semantics. Indeed, if we would take an idea of left merge independently from free merge, then we will provide semantics already expressed by operator of sequential composition. Transition rules for left merge are in the table 2.4.

Table 2.4 Transition rules of Left Merge

$$\frac{\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \surd}{\boldsymbol{x} \mathbin{\lfloor\!\lfloor} \boldsymbol{y} \xrightarrow{\boldsymbol{v}} \boldsymbol{y}} \qquad \frac{\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \boldsymbol{x}'}{\boldsymbol{x} \mathbin{\lfloor\!\lfloor} \boldsymbol{y} \xrightarrow{\boldsymbol{v}} \boldsymbol{x}' \parallel \boldsymbol{y}}$$

From tables 2.1, 2.3 and 2.4 axioms for the free and left merges follow

Table 2.5 Axioms for the Free and Left Merges

FM1	$x \parallel y$	=	$x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x$
LM1	$v \mathbin{⌊\!⌊} y$	=	$v \bullet y$
LM2	$(v \bullet x) \mathbin{⌊\!⌊} y$	=	$v \bullet (x \parallel y)$
LM3	$(x + y) \mathbin{⌊\!⌊} z$	=	$x \mathbin{⌊\!⌊} z + y \mathbin{⌊\!⌊} z$

From the axioms of table 2.5 is clear to see that each term of Process Algebra for parallel processes can be rewritten by equivalent Basic Process Algebra's term.

2.4 Operators of Process Algebra for communicating processes

Process Algebra for communicating processes contain extension of free merge for parallel processes. The new operator we will call merge, or parallel composition. Besides the features of free merge it can execute two process terms simultaneously. For that purpose communication function will be introduced in subsection 2.4.1 , which governs the executions of communication merge. Process algebra for communicating processes is sound and complete (Moller, 1990).

2.4.1 Communication function

The notion that can explain the mechanism of interaction in Process Algebra and outline the issue of communication merge is *communication function*. This function is a partial mapping $\gamma: A \times A \to A$, where A is the set of the atomic actions. It is easier to understand the meaning of the function by the following send/read example.

Send/read communication As follows from the definition the construction of the communication function is possible if there are three atomic actions in the atomic action set, such that we can map two of them on to the third one. In order to fulfil that condition in case of the send/read communication it is enough to have a send action ***s(d),*** a read action ***r(d),*** and a communication action ***c(d)*** for the data ***d*** that corresponds to the send and read actions and expresses the process of information exchange. Communication function is γ ***(s(d), r(d)) = c(d).*** I suppose the function was introduced because we prefer to operate with one process during one step of time and to stay at one state after the execution, that is a standard intention, and the communication function provides the possibility to do it. Indeed, it is impossible to follow several branches of a graph (by which the parallel processes can be represented) simultaneously and to stay at several states.

2.4.2 Communication merge

Communication merge is introduced by following four transition rules.

Table 2.6 Transition rules of Communication Merge

$$\frac{\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \surd \qquad \boldsymbol{y} \xrightarrow{\boldsymbol{w}} \surd}{\boldsymbol{x} \mid \boldsymbol{y} \xrightarrow{\gamma(\boldsymbol{v},\, \boldsymbol{w})} \surd} \qquad \frac{\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \surd \qquad \boldsymbol{y} \xrightarrow{\boldsymbol{w}} \boldsymbol{y}'}{\boldsymbol{x} \mid \boldsymbol{y} \xrightarrow{\gamma(\boldsymbol{v},\, \boldsymbol{w})} \boldsymbol{y}'}$$

$$\frac{\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \boldsymbol{x}' \qquad \boldsymbol{y} \xrightarrow{\boldsymbol{w}} \surd}{\boldsymbol{x} \mid \boldsymbol{y} \xrightarrow{\gamma(\boldsymbol{v},\, \boldsymbol{w})} \boldsymbol{x}'} \qquad \frac{\boldsymbol{x} \xrightarrow{\boldsymbol{v}} \boldsymbol{x}' \qquad \boldsymbol{y} \xrightarrow{\boldsymbol{w}} \boldsymbol{y}'}{\boldsymbol{x} \mid \boldsymbol{y} \xrightarrow{\gamma(\boldsymbol{v},\, \boldsymbol{w})} \boldsymbol{x}' \parallel \boldsymbol{y}'}$$

Communication merge is an extension of communication function. It is clear from the following axioms

$\boldsymbol{a} \mid \boldsymbol{b} = \gamma(\boldsymbol{a},\, \boldsymbol{b})$, if $\gamma(\boldsymbol{a},\, \boldsymbol{b})$ defined, CF1

$\boldsymbol{a} \mid \boldsymbol{b} = \delta$ otherwise. CF2

Here δ is a *deadlock*; that means that the processes have *stopped* its execution and can not proceed.

The other axioms for communication merge are represented below.

Table 2.7 Axioms for Communication Merge

CM1	$\boldsymbol{v} \mid \boldsymbol{w}$	=	$\gamma(\boldsymbol{v},\, \boldsymbol{w})$
CM2	$\boldsymbol{v} \mid (\boldsymbol{w} \bullet \boldsymbol{y})$	=	$\gamma(\boldsymbol{v},\, \boldsymbol{w}) \bullet \boldsymbol{y}$
CM3	$(\boldsymbol{v} \bullet \boldsymbol{x}) \mid \boldsymbol{w}$	=	$\gamma(\boldsymbol{v},\, \boldsymbol{w}) \bullet \boldsymbol{x}$
CM4	$(\boldsymbol{v} \bullet \boldsymbol{x}) \mid (\boldsymbol{w} \bullet \boldsymbol{y})$	=	$\gamma(\boldsymbol{v},\, \boldsymbol{w}) \bullet (\boldsymbol{x} \parallel \boldsymbol{y})$
CM5	$(\boldsymbol{x} + \boldsymbol{y}) \mid \boldsymbol{z}$	=	$\boldsymbol{x} \mid \boldsymbol{z} + \boldsymbol{y} \mid \boldsymbol{z}$
CM6	$\boldsymbol{x} \mid (\boldsymbol{y} + \boldsymbol{z})$	=	$\boldsymbol{x} \mid \boldsymbol{y} + \boldsymbol{x} \mid \boldsymbol{z}$

2.4.3 Merge

As we mentioned, merge is an extension of free merge and is expressed through represented parallel operators by the following axiom.

M1 $\quad x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$

Merge obviously can be depicted as on the Figure 2.6.

$$\begin{tikzcd}
&&&&&& (a\,b) \parallel (b\,a) \arrow[dlllll, "a"'] \arrow[d, "c"] \arrow[drrrrr, "b"] &&&&&& \\
& b \parallel (b\,a) \arrow[dl, "b"'] \arrow[d, "c"] \arrow[drr, "b"] &&&&& b \parallel a \arrow[dl, "b"'] \arrow[d, "c"] \arrow[dr, "a"] &&&&& (a\,b) \parallel a \arrow[dll, "a"'] \arrow[d, "c"] \arrow[dr, "a"] & \\
b\,a \arrow[d, "b"'] & a \arrow[d, "a"'] && b \parallel a \arrow[dl, "b"'] \arrow[d, "c"] \arrow[dr, "a"] && a \arrow[d, "a"'] & \surd & b \arrow[d, "b"'] && b \parallel a \arrow[dl, "b"'] \arrow[d, "c"] \arrow[dr, "a"] && b \arrow[d, "b"'] & a\,b \arrow[d, "a"'] \\
a \arrow[d, "a"'] & \surd & a \arrow[d, "a"'] & \surd & b \arrow[d, "b"'] & \surd && \surd & a \arrow[d, "a"'] & \surd & b \arrow[d, "b"'] & \surd & b \arrow[d, "b"'] \\
\surd && \surd && \surd &&&& \surd && \surd && \surd
\end{tikzcd}$$

Figure 2.6 Process graphs of merge (*a b*) || (*b a*). Here *c* is a communication of two atomic actions from {*a*, *b*}.

2.5 Conclusion

As is apparent from this chapter, Process Algebra has an interesting realisation of the concept of parallelism. Since Basic Process Algebra has analogues for two basic operators of Game Logic, it could rise optimistic prospects in the extension of Game Logic by parallel operators of Process Algebra. However, Process Algebra does not contain a notion of dual game, and its encapsulation would look unnatural. Besides, the possibility of expression of true-false values in Process Algebra looks doubtful. Moreover, it seems that Process Algebra does not support an ideology of two-player game. So, the encapsulation of parallel operators of Process Algebra in the theory of Game Logic is not obvious. However, we will use Process Algebra to discuss some properties which are not represented directly in semantics of tensor and par operators of Linear Logic described in the next chapter. It is analogues of tensor and par operators we will use to extend Game Logic on parallel operators.

3 Parallel games in Linear Logic

As explained, Linear Logic can have a game theoretical interpretation. The semantics given by Blass (1992) relates to Affine Logic. Affine Logic can be obtained from Linear Logic by adding the structural rule of weakening. His semantics was criticised by Abramsky and Jagadeesan (1994), who introduced another game semantics sound and full complete with respect to multiplicative fragment of Linear Logic. Since we consider the games merged by the multiplicative operators as parallel, the conceptual similarity between semantics for additive fragment of Linear Logic and Game Logic allows us to predict the possibility of extension of Game Logic by means of analogues of multiplicative operators of Linear Logic.

3.1 Introduction

In this chapter, the basic features of the games involved by Blass and Abramsky, Jagadeesan are listed in section 3.2 . Since we are going to use the properties of multiplicative operators to enrich the language of Game Logic, in section 3.3 we fulfil in special manner an interpretation of Linear Logic connectives by operations on games. For this purpose we took the basic ideas from Blass and Abramsky, Jagadeesan semantics.

3.2 The game semantics for Linear Logic

Abramsky and Jagadeesan oppose their semantics to that of Blass because they argue with him about the completeness theorem. In any case, we have not need to follow this line because we can not convert a semantics of Linear Logic to the semantics of Game Logic without changes. It is enough to look at the definition of φ^I-*strategy* (Chapter 4), which is used in Game Logic instead of notion of *winning strategy* of Linear Logic. Thus, we can try to make a quintessence of the two semantics and postpone the restrictions on the moment of definition of soundness and completeness inside a model of Game Logic.

3.2.1 The origin of Blass's game semantics for Linear Logic

The Blass's (1992) game semantics for Linear Logic arose over reconsideration of Lorenzen's (1959) game semantics for Propositional and First Order (Classical) Logic. Lorenzen's game semantics includes procedural conventions, which bring asymmetry between players. There is a reason to avoid such structural difference between axiomatics and semantics: logical connectives have a symmetrical status in the axioms and rules of inference, whereas Lorenzen' s game semantics has the asymmetrical conventions. InTable 3.1 there is a brief description of updates, which Blass (1992) offered to avoid the asymmetry of Lorenzen's procedural conventions. One of the asymmetry eliminates by introduction of undetermined games.

Definition 3.1 ***Undetermined games***

We will call a game undetermined if and only if there is no winning strategy for eitherplayer.

Table 3.1 Blass's elimination of the players' asymmetr y

Lorenzen's procedural convention	Blass's transformations
Proponent may only assert an atomic formula after Opponent has asserted it	Every atomic game has to be *undetermined*
Permission or restriction on re-attack (re-defence)	Add extra sort of conjunction (disjunction)

The other asymmetry disappears by using the expressive power of the language of Linear Logic. In Linear Logic there are two sorts of conjunction and disjunction: additive and multiplicative. In Blass's game semantics additive conjunction (disjunction) does not allow to re-attack (re-defence), whereas multiplicatives do allow this.

3.2.2 Basic notions of Game Semantics

There are two basic notions in game semantics: game and winning strategy. We give key moments of their definitions, which we constructed according to the general view of Blass and Abramsky, Jagadeesan. Below we summarise the main properties of the involved games.

1. 2-players: Proponent (P), and Opponent (O)
2. Zero-sum (strictly competitive - either P or O wins)
3. Undetermined atomic games; we can get it by:
   – infinite branches (in Blass (1992) semantics)
   – imperfect information (in Abramsky and Jagadeesan (1994) semantics; they operate with history-independent strategies (a current move depends only from the previous one))

Definition 3.2 ***Game***

A game $\boldsymbol{A}$ depicted by game tree drawn in Figure 3.1 is a quadruple ($\boldsymbol{M_A}$, $\boldsymbol{\lambda_A}$, $\boldsymbol{P_A}$, $\boldsymbol{W_A}$), where

1) $\boldsymbol{M_A} = \{\alpha_0, \alpha_1, \ldots, \alpha_7\}$ - the set of possible moves in game $\boldsymbol{A}$, where α_3, α_5, α_6, α_7 may or may not be the last moves of the game in Figure 3.1. A Greek letter at state on the tree is a name of the last executed move (α_0 is a name for an empty move).

2) $\boldsymbol{\lambda_A} : \boldsymbol{M_A} \rightarrow \{P,O\}$ - labelling function distributing the set of possible moves in game $\boldsymbol{A}$ between the players. A square or circle at a state of the tree shows who has to move at the next turn.

3) $\boldsymbol{P_A}$ - the set of valid positions (bold drawn states of the tree in Figure 3.1) of game $\boldsymbol{A}$. A position denotes the finite unique combination of moves. I.e. at each state we can determine the admissible moves (stressed by bold arrows). The uniqueness of the combination of moves means

that we can not reach a position by the different branches of a tree. That is why the graph has no loops.

4) W_A - the winning branches, by which Proponent can achieve a victory (drawn in dash)

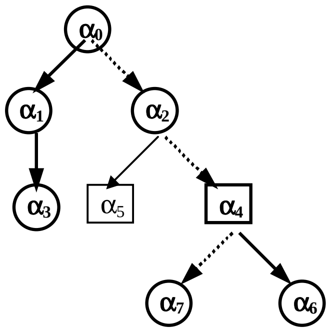


Figure 3.1. A game tree with the set of possible moves {α_0, α_1, ..., α_7}. The set of moves of Proponent is pointed out by squares, of Opponent - by circles. The set of valid positions (admissible moves) stressed by bold, whereas winning branch for Proponent by dash.

Definition 3.3 ***Strategies***

Strategy of a player in game $\boldsymbol{A}$ is a function $\boldsymbol{\sigma}$ into $\boldsymbol{M_A}$, from the set of all positions where the player has to move. The strategy $\boldsymbol{\sigma}$ is a winning one if all branches of σ are in $\boldsymbol{W_A}$.

Eventually, a player can have winning branches and no winning strategy. However, if the player has a winning strategy then she has winning branches.

3.2.3 Winning strategies in Blass' s semantics

Important feature of Blass semantics is an absence of winning strategies for the atomic games. It gives the following results (that you can find out from section 3.3).

1) Games without multiplicative connectives and repetitions do not contain winning strategies for either player. And hence we can not distinguish such games if we would include an outcome of game in game definition
2) If we add a multiplicative part to games then we can obtain winning strategies
3) The known winning strategies are only different sorts of copy strategy
4) When we will refer during following discussion that, for instance, in game $\boldsymbol{A} \vee \boldsymbol{B}$ one of the players has a winning strategy in game $\boldsymbol{A}$, it will mean that $\boldsymbol{A}$ is not an atomic game and contain multiplicative connectives together with a copy strategy for that player
5) *Without copy strategy (even if we suppose winning strategies in atomic formula) multiplicative operators on games would not distinguished from additive one.*

3.3 Game interpretation of the operators of Linear Logic

Since the goal of the introduction of the game semantics is its adaptation to the Game Logic, we will make the following definitions on the special manner that could be considered as transitional.

Definition 3.4 ***Negation***

$\neg \boldsymbol{A} = (\boldsymbol{M_A} , \overline{\boldsymbol{\lambda}}_{\boldsymbol{A}} , \boldsymbol{P_A} , \ \boldsymbol{P_A^\infty} \setminus \boldsymbol{W_A})$

Here we change only scheduling and the goals by switching between two-player.

<u>Winning strategies</u>

A winning strategy (if it exists) of one of the players in game ***A*** becomes the winning strategy of another player in dual game $\neg \boldsymbol{A}$.

Definition 3.5 ***Additive conjunction***

$\boldsymbol{A} \wedge \boldsymbol{B} = (\boldsymbol{M_{A \wedge B}} , \boldsymbol{\lambda_{A \wedge B}} , \boldsymbol{P_{A \wedge B}} , \boldsymbol{W_{A \wedge B}})$

Where:

- $\boldsymbol{M_{A \wedge B}} = \boldsymbol{M_A} + \boldsymbol{M_B} + \{0,1\}$ (if 0 then choose component ***A***, if 1 then choose component ***B***)
- $\boldsymbol{\lambda_{A \wedge B}} = \boldsymbol{\lambda_A} + \boldsymbol{\lambda_B} + (0, O) + (1, O)$, i.e. Opponent choose which game to play
- $\boldsymbol{P_{A \wedge B}}$ is the union of the sets $\boldsymbol{P_A}$ and $\boldsymbol{P_B}$ through prefix position such that:

1) The restriction to moves in $\boldsymbol{M_A}$ (respectively $\boldsymbol{M_B}$) is in $\boldsymbol{P_A}$ (respectively $\boldsymbol{P_B}$)
2) Opponent has to choose in the prefix position which game to play

- $\boldsymbol{W_{A \wedge B}}$ is the union of the sets $\boldsymbol{W_A}$ and $\boldsymbol{W_B}$ through prefix position

<u>Winning strategies</u>

Proponent has a winning strategy in compound game if she has winning strategies in both components. Opponent has a winning strategy if he has it in one of the components.

Definition 3.6 ***Additive disjunction***

$\boldsymbol{A} \vee \boldsymbol{B} = \neg (\neg \boldsymbol{A} \wedge \neg \boldsymbol{B})$

It gives the switching of roles between Opponent and Proponent

<u>Winning strategies</u>

Proponent has a winning strategy in compound game if she has winning strategies in one of the components. Opponent has a winning strategy if he has it in both components.

The general significant difference of semantics' description for multiplicative connectives compared to the additive one is that we will use multi-states to express a compound game.

Definition 3.7 ***Multiplicative conjunction (Tensor)***

Tensor is the game combination $\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{M_{A \otimes B}} , \boldsymbol{\lambda_{A \otimes B}} , \boldsymbol{P_{A \otimes B}} , \boldsymbol{W_{A \otimes B}})$, where:

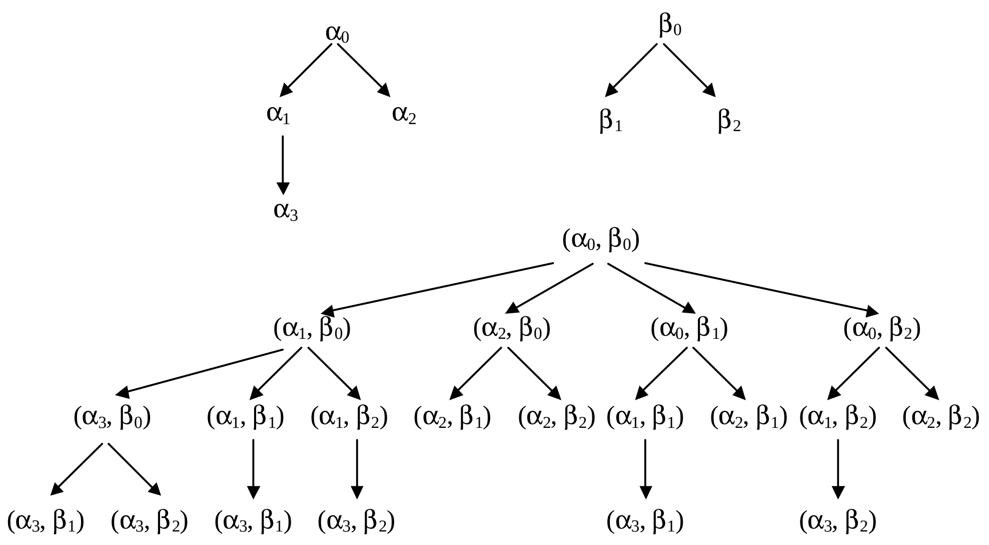


Figure 3.2. The set of all possible moves in compound game with a multiplicative connective is the following partial Cartesian product of the two component games

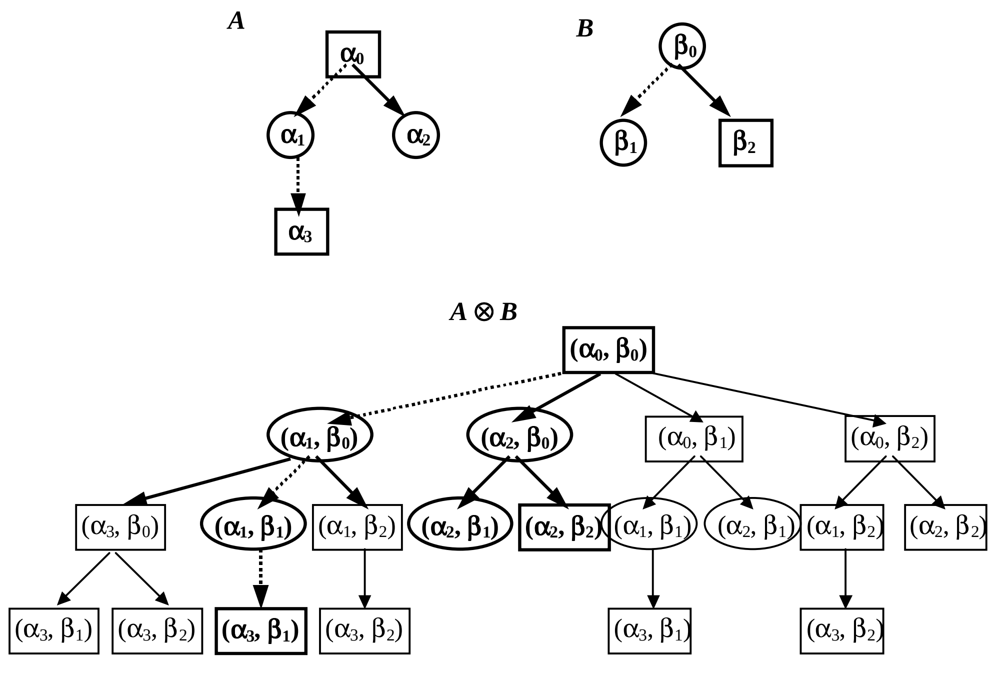


Figure 3.3 The way by which parameters of the two component games are transformed ir the resulting tensor game

- $\boldsymbol{M}_{\boldsymbol{A} \otimes \boldsymbol{B}}$ is the partial Cartesian product of two possible moves' sets $\boldsymbol{M}_{\boldsymbol{A}}$ and $\boldsymbol{M}_{\boldsymbol{B}}$, as is shown in Figure 3.2. It is constructed as follows. Take the initial states of the component games (proto-states) and make from them an initial complex state. Construct the complex states of the next level by fixing consecutively each of the proto-state of initial complex state and taking successors of another one from the component game. By applying that procedure to each complex state we can construct a multiplicative tree of the set of possible moves.
- $\boldsymbol{\lambda}_{\boldsymbol{A} \otimes \boldsymbol{B}}$ labelling function: it is Opponent's turn to move if and only if it is her turn to move in both games, otherwise it is Proponent's turn to move
- $\boldsymbol{P}_{\boldsymbol{A} \otimes \boldsymbol{B}}$ is the set of valid positions (admissible moves) that constructed as follows:
  1) If it is Opponent's turn to move she can move in any possible direction (during her turn she is allowed to move in both components, and hence she can always switch the game)
  2) If it is Proponent's turn to move he can move only in the component where it is his turn to move (note that if he has turn to move in both components, he can switch the games)
- $\boldsymbol{W}_{\boldsymbol{A} \otimes \boldsymbol{B}}$ is the set of all winning for Proponent branches that equal to intersection of the winning branches of the component games (i.e. Proponent has to win in both components of the resulting multiplicative branch).

Winning strategies

Proponent has a winning strategy in compound game if he has winning strategies in both components. Opponent has a winning strategy if she has it in one of the components. Besides, Opponent can have a winning strategy in compound game even if nobody has winning strategies in component games and, at the same time, she can apply one of the copy strategies. For instance, a copy strategy can be applied by Opponent (she can switch between two games and hence to copy moves from one to another) if we have a compound game $\boldsymbol{A} \otimes \neg \boldsymbol{A}$. Another opportunity of applying a winning strategy for Opponent is a game $\boldsymbol{A} \otimes \boldsymbol{A}$ over nonprincipal ultrafilter on the set of natural numbers (see, for instance, Blass 1992).

Definition 3.8 *Multiplicative disjunction (Par)*

Par – is the game combination $\boldsymbol{A} \oplus \boldsymbol{B} = \neg (\neg \boldsymbol{A} \otimes \neg \boldsymbol{B})$, where (see Figure 3.3):

- $\boldsymbol{M}_{\boldsymbol{A} \oplus \boldsymbol{B}} = \boldsymbol{M}_{\boldsymbol{A} \otimes \boldsymbol{B}}$
- $\boldsymbol{\lambda}_{\boldsymbol{A} \oplus \boldsymbol{B}}$ labelling function: Proponent has to move if and only if he has to move in both games, otherwise Opponent has to move
- $\boldsymbol{P}_{\boldsymbol{A} \oplus \boldsymbol{B}}$ is the set of valid positions (admissible moves) constructed as follows:
  1) If it is Proponent's turn to move he can move in any possible direction (during his turn he is allowed to move in both components, and hence he can always switch the game)
  2) If it is Opponent's turn to move she can move only in the component where it is her turn to move (note that if it is her turn to move in both components, she can switch the games)

- $W_{A \oplus B}$ is the set of all winning for Proponent branches that equal to union of the winning branches of the component games (i.e. for Proponent enough to win in one of the components of the resulting multiplicative branch).

<u>*Winning strategies*</u>

Proponent and Opponent switch the roles in comparison with the tensor games.

3.4 Games with several multiplicative operators

The multiplicative operators yield complex states that were considered in previous section. It can give rise to the following questions:

– how to determine who has to move in such games,

– in which components the player can switch.

The first question has quite an evident answer: we have to count from the lowest level of sub-games whose turn it is to move according to atomic game schedule and multiplicative connectives. An example of such a counting is depicted in Figure 3.4.

The next question is easier to find out now we have an answer to the first one. A player can make a switch to those atomic games where both she/he has a move and in each subformula of main formula that include this game she/he also has to move. See Figure 3.5.

$$\begin{array}{rc}
\text{Game:} & (A \oplus \neg A) \otimes (B \oplus \neg B) \\
\lambda_A, \lambda_{\neg A}, \lambda_B, \lambda_{\neg B}: & \underbrace{\underbrace{\text{O} \quad \text{P}}_{\text{O}} \qquad \underbrace{\text{P} \quad \text{O}}_{\text{O}}}_{\text{O}} \\
\lambda_{A \oplus \neg A}, \lambda_{B \oplus \neg B}: & \\
\lambda_{(A \oplus \neg A) \otimes (B \oplus \neg B)}: &
\end{array}$$

Figure 3.4 Labelling function for games with several multiplicative operators

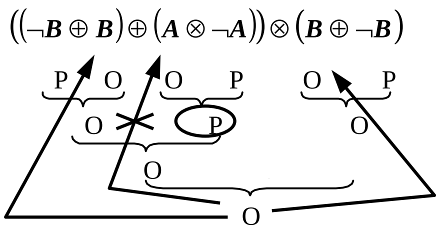


Figure 3.5 Admissible atomic plays for switcher

4 An extension of Game Logic with Parallel Operators

Linear Logic has two-player game semantics, and contains multiplicative operators: tensor and par. Game Logic arose from Propositional Dynamic Logic and was introduced by Parikh (1985) as a language to explore games.

The language of Game Logic has two constituents, formulas (propositions) and games. A set of atomic games Π_0 and a set of atomic formulas Φ_0 are at the basis over which we can define games π and formulas φ (Pauly, 1999):

$\pi := g \mid \varphi ? \mid \pi ; \pi \mid \pi \cup \pi \mid \pi^* \mid \pi^d$

$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle \varphi$

where $p \in \Phi_0$ and $g \in \Pi_0$. Also the following equations are valid

$\top := \neg\bot$,

$[\pi]\varphi := \neg\langle\pi\rangle \neg\varphi$,

$\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$,

$\varphi \rightarrow \psi := \neg\varphi \vee \psi$.

In the chapter we introduce semantics, syntax and axiomatics of parallel operators for Game Logic, basing on definitions for multiplicative operators of Linear Logic.

4.1 Preliminaries

This section contains general mathematical definitions, which will be used in this chapter.

Multiset is a *set*-like object in which order is ignored, but multiplicity is explicitly significant. Therefore, multisets {1, 2, 3} and {3, 1, 2} are equivalent, but {1, 2, 3} and {1, 1, 2, 3} differ. Given a set *S*, the ***power set*** of *S* is the set of all subsets of *S*. The order of a power set of a set of order n is 2^n. The power set of *S* can be denoted as 2^S or *P*(*S*).

List is a data structure consisting of an ordered *set* of elements, each of which may be a number, another list, etc. A list is usually denoted $(a_1, a_2, ..., a_n)$ or $\langle a_1, a_2, ..., a_n\rangle$, and may also be interpreted as a vector. Multiplicity matters in a list, so (1, 1, 2) and (1, 2) are not equivalent. ***Proper List*** is a prefix of list, which is not the entire list. For example, consider a list ⟨1, 2, 3, 4⟩. Then ⟨1, 2, 3⟩ and ⟨1⟩ are proper subsets, while ⟨1, 2, 3, 4⟩and ⟨1, 2, 5⟩ are not. If *p* and *q* are the lists, then $q < p$ will denote that *q* is a proper list of *p*. The notation $q \leq p$ is conventionally used to denote that $q = p$, or $q < p$. A list *x* of the set of lists *S* is a ***maximal list of the set of lists*** if and only if it is not a proper list of any list from the set.

A ***tree*** is a mathematical structure, which can be viewed as either a graph or as a data structure. The two views are equivalent, since a tree data structure contains not only a set of elements, but also connections between elements, making up a tree graph. A tree graph is a set of straight-line segments connected at their ends containing no closed loops (cycles). In other words, it is a simple, undirected, connected, acyclic graph. A tree with n nodes has n - 1 edges. Conversely, a connected graph with n nodes and n - 1 edges is a tree. ***Maximal Sub-Tree for given node of tree*** is a sub-tree that contains all sub-branches, starting from the given node of the initial tree.

4.2 Game model and game structures

In this section we define a game structure. It is a formal general definition of game. The definition is adapted to application with the parallel operators. For this purpose we involved specified mask function. Besides, we discuss the differences between *winning strategy* and φ^I*-strategy*, which is an extension of winning strategy for a game model of Game Logic. At the end of the section we define the game model for Game Logic with Parallel Operators.

4.2.1 Game structure

We consider the games between Proponent (**P**) and Opponent (**O**), and define a game tree as a following *game structure.*

Definition 4.1 ***Game structure***

A game structure $\boldsymbol{G}$ is defined by the tuple ($\boldsymbol{S}$, $\boldsymbol{M}$, χ, λ, $\boldsymbol{R}$,μ^P, μ^O), where

1) $\boldsymbol{S} = \{s_0, s_1, \ldots, s_n\}$ is a finite set of *states*
2) $\boldsymbol{M} = \{m_0, m_1, \ldots, m_k\}$is a finite set of all possible *atomic moves*
3) $\boldsymbol{R} = \{q^0, q^1, \ldots, q^h\}$ is a set of *admissible sequences of moves* (*histories, or branches*) ; a sequence of moves is a list

   If $\mathbf{q} \in \boldsymbol{R}$ and $\mathbf{p} \le \mathbf{q}$ *then* $\mathbf{p} \in \boldsymbol{R}$
4) χ: $\boldsymbol{R} \to \boldsymbol{S}$ is a *trace* function defining a state that we can achieve by fixed history
5) λ: $\boldsymbol{S} \to \{\mathbf{P}, \mathbf{O}, \mathbf{E}\}$ is a *scheduling* function, which determines for each state who has to move at the next turn; for all final states *f*, which have no successors: $\lambda(f)= E$
6) μ^P: $\boldsymbol{S} \to \{\top, \bot\}$ is a *specified mask* function for Proponent
7) μ^O: $\boldsymbol{S} \to \{\top, \bot\}$ is a specified mask function for Opponent

A part of a sequence of moves we will call a ***segment*** if only one player has moves inside this part.

We may define an *atomic game.* In spite of atomic statements used in Logic and atomic moves in Game Theory, atomic games have an internal structure, which nevertheless is simplest among the other types of games. The structure has an influence on the winning strategies of the players. *An atomic game is considered to contain only such game structures for which specified mask function of Proponent coincides with that of Opponent.* The statement is true also for game structures of all non-parallel games. The reason for this we explain in the third clause of remark to the Definition 4.14 for tensor game.

Example of a game tree.

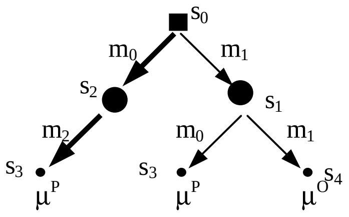


Figure 4.1 A game tree. The set of states is represented by circles (with Opponent turn to move), squares (with Proponent turn to move), and points (nobody has to move). Arrows represent the set of moves. Bold arrows show Proponent's φ^I*-strategy*.

In Figure 4.1 we depict a general example of game tree. Game structure in this case has the following content.

1) $\boldsymbol{S} = \{s_0, s_1, s_2, s_3, s_4\}$
2) $\boldsymbol{M} = \{m_0, m_1, m_2\}$
3) $\boldsymbol{R} = \{\langle\,\rangle, \langle m_0\rangle, \langle m_1\rangle, \langle m_0, m_2\rangle, \langle m_1, m_0\rangle, \langle m_1, m_1\rangle\}$
4) $\chi(\langle\,\rangle) = s_0$; $\chi(\langle m_0\rangle) = s_2$; $\chi(\langle m_1\rangle) = s_1$; $\chi(\langle m_0, m_2\rangle) = s_3$; $\chi(\langle m_1, m_0\rangle) = s_3$; $\chi(\langle m_1, m_1\rangle) = s_4$
5) $\lambda(s_0) = \mathbf{P}$; $\lambda(s_1) = \mathbf{O}$; $\lambda(s_2) = \mathbf{O}$; $\lambda(s_3) = \mathbf{E}$; $\lambda(s_4) = \mathbf{E}$
6) $\mu^P(s_3) = \top$; $\mu^P(s_0) = \bot$; $\mu^P(s_1) = \bot$; $\mu^P(s_2) = \bot$; $\mu^P(s_4) = \bot$
7) $\mu^O(s_4) = \top$; $\mu^O(s_0) = \bot$; $\mu^O(s_1) = \bot$; $\mu^O(s_2) = \bot$; $\mu^O(s_3) = \bot$

Game structure can be applied to the different types of game. Later on we will present axioms for given semantics, but the axioms are not valid for all types of game. Below we express necessary types through the given game structure.

Besides, during further discussion, we will call a taken sequence of moves $\boldsymbol{x}$, such that $\exists \mathbf{p}$: $(\mathbf{p}\boldsymbol{x} \in \boldsymbol{R}_\pi) \wedge (\chi_\pi(\mathbf{p}) = s) \wedge (\lambda_\pi(\chi_\pi(\boldsymbol{x})) = \mathbf{E})$, a ***play*** of game π from given state s.

Definition 4.2 φ^I***-zero-sum game***

A game π is a φ^I-zero-sum from given state $s = \chi_\pi(\mathbf{p})$ if and only if

$\forall\, \mathbf{q} \in \boldsymbol{R}_\pi\, (((\mathbf{p} \le \mathbf{q}) \wedge (\lambda(\mathbf{q}) = \mathbf{E})) \rightarrow$

$(\exists \mathbf{r} \in \boldsymbol{R}_\pi\, ((\mathbf{p} \le \mathbf{r}) \wedge (\mathbf{r} \le \mathbf{q}) \wedge ((\mu^P{}_\pi(\chi_\pi(\mathbf{r})) = \top) \vee (\mu^O{}_\pi(\chi_\pi(\mathbf{r})) = \top)))))$

Remark. ***Why*** φ^I***-zero-sum***

We call such type of games φ^I -zero-sum because it seems to use an ideology of zero sum game taken from the games which use a terminology of winning strategies and applied to the games with language of φ^I-*strategy* (see Definition 4.7 below). The idea is that two-player can not loose or win simultaneously, but may have a draw. In our case, we consider reaching a state where φ is valid for either player (or in other words, that function μ is determined for both players) as a draw that is quite naturals. Therefore we follow the idea of zero-sum games.

Definition 4.3 ***Determined and undetermined games***

A game π is *determined* from given state *s* if and only if it is possible, starting from that state, to present a φ^I-*strategy* (see Definition 4.7 below) for one player or a ψ^I-*strategy* for the other ($\psi=\neg\varphi$). Otherwise the game is *undetermined* from given state *s*.

4.2.2 Winning branch and winning strategy

The notion of winning strategy plays key role in the semantics of game theory. For given game structure γ it is possible to determine a set of winning states or a set of winning sequences of moves for each player: $\boldsymbol{W}_\gamma^{\mathrm{P}}$, $\boldsymbol{W}_\gamma^{\mathrm{O}}$. Player can achieve a winning position (state) of a game under the influence of different reasons. If it happens because her/his competitor made a mistake during the game, then we can say that the winner has a *winning branch*, which he followed at the game. If the player can achieve a winning position irrespective of mistakes of the competitor then we can say that the player has a*winning strategy.*

Game structure, defined in subsection 4.2 , allows us to play games not only from the initial state; for instance, in chess we have an opportunity to decide endgame problems. It can be useful, because even if we could not find a winning strategy for either player in full chess game, we can find a winning strategy in most of chess endgames. Therefore, below we give precise definitions of the ‘strategic’ notions, which can be considered from the given state of a game.

Definition 4.4 ***Winning branch for the given state s***

Player $\mathbf{N}\in\{\mathbf{O}, \mathbf{P}\}$ has a winning branch in the game γ starting from state *s* if and only if there is a sequences of moves **p** in game γ such that $s = \chi(\mathbf{p})$ and there is $\boldsymbol{x}$ such that $\langle \mathbf{p}\boldsymbol{x} \rangle \in \boldsymbol{W}_\gamma^{\mathbf{N}}$.

Definition 4.5 ***Winning strategy for the given state s***

Player $\mathbf{N}\in\{\mathbf{O}, \mathbf{P}\}$ has a winning strategy in game γ starting from state *s* if and only if

$\exists\, \mathbf{p} \in \boldsymbol{R}\, (\chi(\mathbf{p})=s \wedge \exists \mathbf{T} = \{\boldsymbol{x}: \langle \mathbf{p}\boldsymbol{x}\rangle \in \boldsymbol{R}_\gamma \;\wedge\; \forall\, \boldsymbol{y} < \boldsymbol{x}\, (\lambda(\chi(\langle \mathbf{p}\boldsymbol{y}\rangle)) = \mathbf{N} \;\rightarrow\; (\exists \mathrm{m}_1, \mathrm{m}_2 \in \boldsymbol{M}_\gamma\, \forall\, \boldsymbol{z}_1, \boldsymbol{z}_2\, ((\langle \boldsymbol{y}\mathrm{m}_1\boldsymbol{z}_1\rangle \in \boldsymbol{R} \vee \langle \boldsymbol{y}\mathrm{m}_2\boldsymbol{z}_2\rangle \in \boldsymbol{R}) \rightarrow \boldsymbol{x} = (\langle \boldsymbol{y}\mathrm{m}_1\boldsymbol{z}_1\rangle \vee \langle \boldsymbol{y}\mathrm{m}_2\boldsymbol{z}_2\rangle))))\}\; \forall \boldsymbol{x}\, ((\boldsymbol{x}\in \mathbf{T} \wedge \lambda(\chi(\langle \boldsymbol{x}\rangle)) = \mathbf{E}) \rightarrow \langle \mathbf{p}\boldsymbol{x}\rangle \in \boldsymbol{W}_\gamma^{\mathbf{N}}))$

The subtree **T** is a winning strategy for the given state.

If a game has a winning strategy for given state *s*, Definition 4.6 gives description of it in terms of subtree that corresponds to the winning strategy. The definition also shows whether the winning strategy exists. The subtree of winning strategy can be obtained from the tree that corresponds to the game starting from state *s* by eliminating some (but not all for each node) of its subtrees that start with moves of player **N**. All of the remaining outcomes of the constructed subtree have to be in the striving for **N** states.

Consider apart the formula given in Definition 4.6. We begin a game from state $s = \chi(\mathbf{p})$ that has to be reached by admissible sequence of move $\mathbf{p}$. We are looking for the tree $\mathbf{T}$ that starts from state s and copies all admissible sequences of moves of subtree $\langle \mathbf{p}\boldsymbol{x}\rangle \in \boldsymbol{R}_\gamma$ except in those which are the alternatives for the player $\mathbf{N}$ moves $\lambda(\chi(\langle \mathbf{p}\boldsymbol{y}\rangle)) = \mathbf{N} \rightarrow (\exists m_1, m_2 \in \boldsymbol{M}_\gamma \, \forall \, \boldsymbol{z}_1, \boldsymbol{z}_2 \, (\langle y m_1 \boldsymbol{z}_1\rangle \in \boldsymbol{R} \vee \langle y m_2 \boldsymbol{z}_2\rangle \in \boldsymbol{R}) \rightarrow \boldsymbol{x} = (\langle y m_1 \boldsymbol{z}_1\rangle \vee \langle y m_2 \boldsymbol{z}_2\rangle))$. Besides, all maximal sequences of moves of the tree has to be in winning set of player $\mathbf{N}$: $\forall \boldsymbol{x} \, ((\boldsymbol{x} \in \mathbf{T} \wedge \lambda(\chi(\langle \boldsymbol{x}\rangle)) = \mathbf{E}) \rightarrow \langle \mathbf{p}\boldsymbol{x}\rangle \in \boldsymbol{W}_\gamma^{\mathbf{N}}))$.

4.2.3 φ^I-strategy

For application in game models we have to extend the notion of winning strategy to the notion of φ^I-strategy. But first we determine the notion of X-strategy.

Definition 4.6 ***X- strategy for the given state s***

Player $\mathbf{N} \in \{\mathbf{O}, \mathbf{P}\}$ has an X-strategy in game γ starting from the state s if and only if it is possible to construct a *winning* strategy for the player from the state s, which all outcome states would belong to X, where X is a set of states.

Definition 4.7 ***φ^I- strategy for the given state s***

Player $\mathbf{N} \in \{\mathbf{O}, \mathbf{P}\}$ has an φ^I-strategy in game γ starting from the state s for model I if and only if she/he has an X-strategy, and for all states $s \in X$ and all their successor states (if their exists): $I, s \vDash \varphi$.

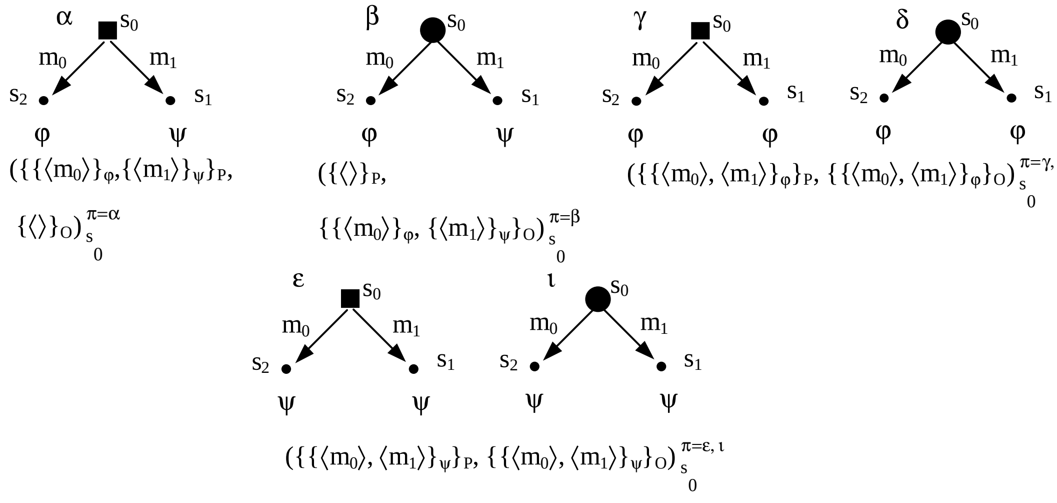
Notion of φ^I-strategy plays key role in definition of game model and hence in validity of axioms. Therefore, consider how φ^I-strategy can occur at the games.

Each player can have in a game π φ^I-strategy, or ψ^I-strategy (where $\psi = \neg\varphi$), or both of them, or nothing.

Definition 4.8 ***Strategy couple***

Determine a ***strategy couple*** as $(X, Y)^{\pi}_{s_0}$, where X is a set of strategies which has Proponent in game π, starting from the state s_0 (namely φ^I-strategy, or ψ^I-strategy (where $\psi = \neg\varphi$), or both of them, or nothing), and Y – for Opponent.

However, not all of the combinations are represented in given types of games. Thus we have to examine all possible cases.



α
β
γ
δ
ε
ι
s_0
m_0
m_1
s_1
s_2
φ
ψ
$(\{\{\langle m_0\rangle\}_\varphi,\{\langle m_1\rangle\}_\psi\}_P, \{\langle\rangle\}_O)^{\pi=\alpha}_{s_0}$
$(\{\langle\rangle\}_P, \{\{\langle m_0\rangle\}_\varphi, \{\langle m_1\rangle\}_\psi\}_O)^{\pi=\beta}_{s_0}$
$(\{\{\langle m_0\rangle, \langle m_1\rangle\}_\varphi\}_P, \{\{\langle m_0\rangle, \langle m_1\rangle\}_\varphi\}_O)^{\pi=\gamma,\delta}_{s_0}$
$(\{\{\langle m_0\rangle, \langle m_1\rangle\}_\psi\}_P, \{\{\langle m_0\rangle, \langle m_1\rangle\}_\psi\}_O)^{\pi=\varepsilon,\iota}_{s_0}$

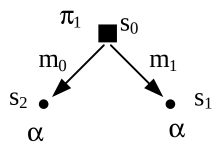
Figure 4.2 Occurrences of φ-strategy and ψ-strategy (where ψ=¬φ) in a game when the game starts from the last node.

Definition 4.9 *Shadow of strategy couple*

We will call a Boolean quadruple ***shadow of strategy couple*** if it is the following projection of the strategy couple:

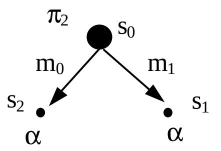
- first element of tuple is true if Proponent has a φ^I-strategy; otherwise it false,
- second element is true if Proponent has a ψ^I-strategy; otherwise it false,
- third element is true if Opponent has a φ^I-strategy; otherwise it false,
- fourth element is true if Opponent has a ψ^I-strategy; otherwise it false.

First, consider the cases when a game starts from the last node. We assume that in each state of the game either function φ is valid or its negation, but not both of them. We supposed that ψ=¬φ; then we have to check only the combinations of states where the functions are valid and of the players moves as it is depicted in Figure 4.2. In game α Proponent has φ^I-strategy $\{\langle m_0\rangle\}_\varphi$, and ψ^I-strategy $\{\langle m_1\rangle\}_\psi$, whereas Opponent has none (the shadow of strategy couple for this game is $(\top, \top, \bot, \bot)$). In game β the picture is opposite (the shadow of strategy couple for this game is $(\bot, \bot, \top, \top)$). Games γ and δ (as ε and ι) are equal in this sense. In γ Proponent has φ^I-strategy $\{\langle m_0\rangle, \langle m_1\rangle\}_\varphi$ and Opponent has the same (the shadow of strategy couple for this game is $(\top, \bot, \top, \bot)$). In ε Proponent has ψ^I-strategy $\{\langle m_0\rangle, \langle m_1\rangle\}_\psi$ and Opponent has the same (the shadow of strategy couple for this game is $(\bot, \top, \bot, \top)$).



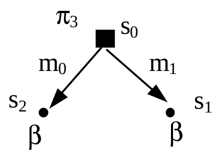
$(\{\{\langle m_0\rangle X_{\varphi\alpha}, \langle m_1\rangle X_{\varphi\alpha}\}_{\varphi}$

$\{\langle m_0\rangle X_{\psi\alpha}, \langle m_1\rangle X_{\psi\alpha}\}_{\psi}\}_{P}, \{\langle\rangle\}_{O})^{\pi_1}_{s_0}$



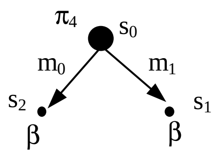
$(\{\{\langle m_0\rangle X_{\varphi\alpha}, \langle m_1\rangle X_{\varphi\alpha}\}_{\varphi}$

$\{\langle m_0\rangle X_{\psi\alpha}, \langle m_1\rangle X_{\psi\alpha}\}_{\psi}\}_{P}, \{\langle\rangle\}_{O})^{\pi_2}_{s_0}$



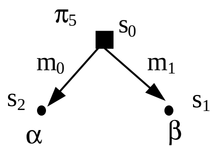
$(\{\langle\rangle\}_{P}, \{\{\langle m_0\rangle Y_{\varphi\beta}, \langle m_1\rangle Y_{\varphi\beta}\}_{\varphi}$

$\{\langle m_0\rangle Y_{\psi\beta}, \langle m_1\rangle Y_{\psi\beta}\}_{\psi}\}_{O})^{\pi_3}_{s_0}$



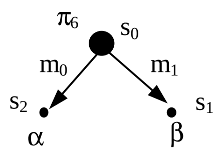
$(\{\langle\rangle\}_{P}, \{\{\langle m_0\rangle Y_{\varphi\beta}, \langle m_1\rangle Y_{\varphi\beta}\}_{\varphi}$

$\{\langle m_0\rangle Y_{\psi\beta}, \langle m_1\rangle Y_{\psi\beta}\}_{\psi}\}_{O})^{\pi_4}_{s_0}$



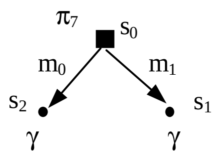
$(\{\{\langle m_0\rangle X_{\varphi\alpha}\}_{\varphi}\{\langle m_0\rangle X_{\psi\alpha}\}_{\psi}\}_{P},$

$\{\langle\rangle\}_{O})^{\pi_5}_{s_0}$



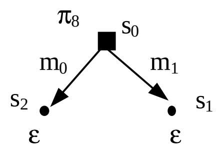
$(\{\langle\rangle\}_{P},$

$\{\{\langle m_1\rangle Y_{\varphi\beta}\}_{\varphi}\{\langle m_1\rangle Y_{\psi\beta}\}_{\psi}\}_{O})^{\pi_6}_{s_0}$



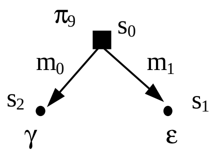
$(\{\{\langle m_0\rangle X_{\varphi\gamma}, \langle m_1\rangle X_{\varphi\gamma}\}_{\varphi}\}_{P},$

$\{\{\langle m_0\rangle Y_{\varphi\gamma}, \langle m_1\rangle Y_{\varphi\gamma}\}_{\varphi}\}_{O})^{\pi_7}_{s_0}$



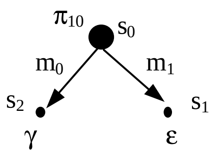
$(\{\{\langle m_0\rangle X_{\psi\varepsilon}, \langle m_1\rangle X_{\psi\varepsilon}\}_{\psi}\}_{P},$

$\{\{\langle m_0\rangle Y_{\psi\varepsilon}, \langle m_1\rangle Y_{\psi\varepsilon}\}_{\psi}\}_{O})^{\pi_8}_{s_0}$



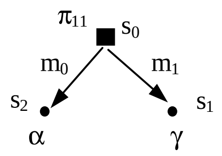
$(\{\{\langle m_0\rangle X_{\varphi\gamma}\}_{\varphi}, \{\langle m_1\rangle X_{\psi\varepsilon}\}_{\psi}\}_{P},$

$\{\langle\rangle \}_{O})^{\pi_9}_{s_0}$



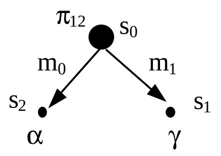
$(\{\langle\rangle\}_{P}, \{\{\langle m_0\rangle Y_{\varphi\gamma}\}_{\varphi},$

$\{\langle m_1\rangle Y_{\psi\varepsilon}\}_{\psi} \}_{O})^{\pi_{10}}_{s_0}$



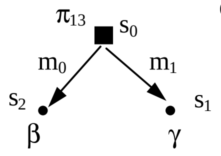
$(\{\{\langle m_0\rangle X_{\varphi\alpha}, \langle m_1\rangle X_{\varphi\gamma}\}_{\varphi},$

$\{\langle m_0\rangle X_{\psi\alpha}\}_{\psi} \}_{P}, \{\langle\rangle\}_{O})^{\pi_{11}}_{s_0}$



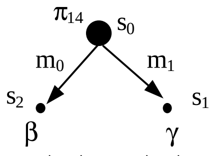
$(\{\langle m_1\rangle X_{\varphi\gamma}\}_{\varphi}\}_{P},$

$\{\{\langle m_1\rangle Y_{\varphi\gamma}\}_{\varphi} \}_{O})^{\pi_{12}}_{s_0}$



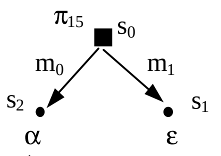
$(\{\langle m_1\rangle X_{\varphi\gamma}\}_{\varphi}\}_{P},$

$\{\{\langle m_1\rangle Y_{\varphi\gamma}\}_{\varphi} \}_{O})^{\pi_{13}}_{s_0}$



$(\{\langle\rangle\}_{P}, \{\{\langle m_0\rangle Y_{\varphi\beta}, \langle m_1\rangle Y_{\varphi\gamma}\}_{\varphi},$

$\{\langle m_0\rangle Y_{\psi\beta}\}_{\psi}\}_{O})^{\pi_{14}}_{s_0}$



$(\{\{\langle m_0\rangle X_{\varphi\alpha}\}_{\varphi},$

$\{\langle m_0\rangle X_{\psi\alpha}, \langle m_1\rangle X_{\psi\gamma}\}_{\psi}\}_{P}, \{\langle\rangle\}_{O})^{\pi_{15}}_{s_0}$

Figure 4.3 Occurrences of φ-strategy and ψ-strategy (where ψ=¬φ) in a game when the game starts not from the last node.

Next, examine the general case. Consider the nodes, which lead to the games with shadows of strategy couples as we had in Figure 4.2. Let us check whether any combination of such games gives a new shadow of strategy couple. In Figure 4.3 we draw up a sufficient amount of graphs to see that any combination of games does not give a new shadow of strategy couple[*]. For instance in game π_1 Proponent has a φ^I-strategy $\{\langle m_0\rangle X_{\varphi\alpha}, \langle m_1\rangle X_{\varphi\alpha}\}_{\varphi}$ and ψ^I-strategy $\{\langle m_0\rangle X_{\psi\alpha}, \langle m_1\rangle X_{\psi\alpha}\}_{\psi}$ whereas Opponent has none; hence, the shadow of strategy couple for this game is $(\top, \top, \bot, \bot)$ which is equal to the shadow of strategy couple for game α. It is easy to see that we can not obtain a new shadow of strategy couple without special assumptions. Now we can make the following claim.

Claim 4.1 ***Shadow of strategy couple for determined games***

The shadows of strategy couple for finite determined games have the following values:

1) $(\top, \top, \bot, \bot)$

2) $(\bot, \bot, \top, \top)$

3) $(\top, \bot, \top, \bot)$

4) $(\bot, \top, \bot, \top)$

Obviously we can not have a shadow like $(\top, \top, \bot, \top)$ or $(\top, \top, \top, \top)$ anytime, because it is impossible that Opponent would have a ψ^I-strategy whereas Proponent has a φ^I-strategy. Indeed, the fact that one player has a strategy to achieve a goal entails that the other player can not achieve an opposite goal.

However, the four represented cases are not enough to express the features of parallel operators (section 4.3.2) that we will discuss in subsection Claim 4.2. The reason is a copy strategy that players do not need if they have any other strategy in the playing game. Thus, to have a reason to apply a copy strategy we have to find a way of appearance of shadow in form $(\bot, \bot, \bot, \bot)$.

Such a shadow can appear in infinite games, like it is shown in Figure 4.4 game α, and in games with *imperfect information* shown in game β. In such games we can also obtain a shadow for which only one element is true. In case of α, if we assume that s_2 is final state and φ is valid in the state then we obtain the shadow of strategy couple $(\top, \bot, \bot, \bot)$. In case of β, if we predetermine that in state s_4 φ is valid then we obtain the shadow of strategy couple $(\bot, \bot, \top, \bot)$.

[*] Only for the Figure 4.3, dots in the graphs are not the final states, but correspond to the starting states of the games pointed out in the leaves.

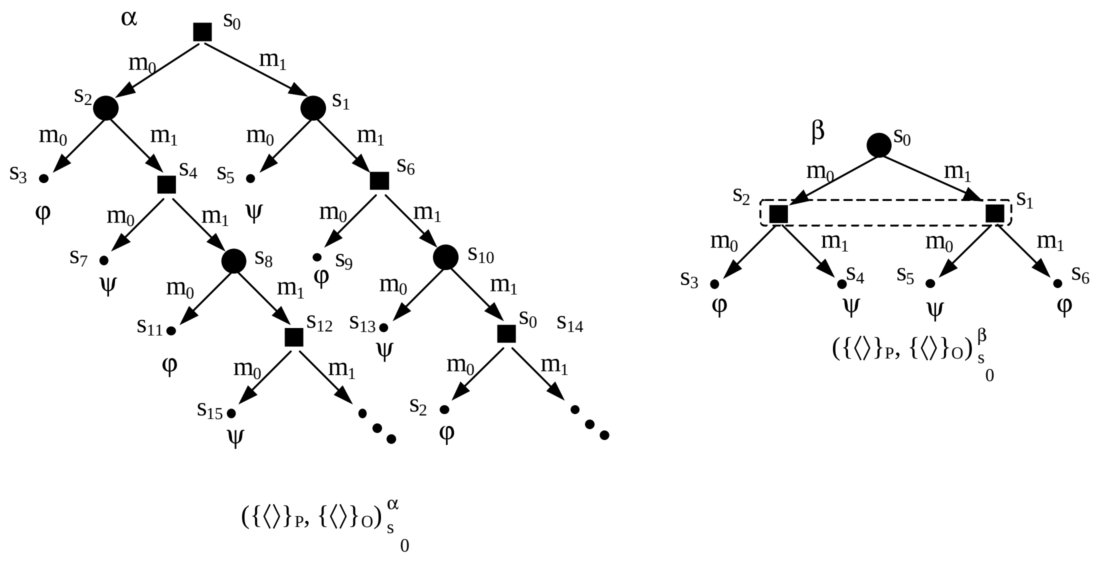


Figure 4.4 The cases when the *shadow of strategy couple* is equal to (⊥, ⊥, ⊥, ⊥).

Claim 4.2 ***Shadow of strategy couple for undetermined games***

The shadows of strategy couple for undetermined games can be equal to one of those of determined games or one of the following:

1) $(\top, \top, \bot, \bot)$

2) $(\bot, \bot, \top, \top)$

3) $(\top, \bot, \top, \bot)$

4) $(\bot, \top, \bot, \top)$

4.2.4 The role of copy strategy in model distinction of operators

In this subsection all discussion will concern φ^I-zero-sum games for parallel operator, which are played from the given state. It does not mean that the games, which constitute a parallel game, have to be φ^I-zero-sum. That would be confirmed by the Figure 4.6 if you will predetermine function μ^P or μ^O for state s_2 of game β.

First of all consider the influence of the assumption that the game structure is determined on the semantics of parallel operators. If a game is determined then for a given game we can say in advance whether a player has a φ^I-strategy for this game or not. This means that we do not need to appeal to the peculiarity of parallel operators that lies in switching between the games to find a φ^I-strategy. If a player knows in which game she/he has a φ^I-strategy, she/he can choose the game immediately. Furthermore, it will mean that we lose all semantic distinctions in game model between operators of choice $\cap$, $\cup$ and the parallel operators $\otimes$, $\oplus$ respectively. The opportunity to leave one of the two component games of a parallel operator without playing gives rise to the equivalence.

If we suppose that the involved games are undetermined then the search of φ^I-strategy for choice operators reminds us of a Russian Roulette: when pulling the trigger you do not know what you will get either a bullet or a Muscovite maiden. On the contrary, switching function of parallel operators gives a chance to impress the Muscovite maiden without being shot through brains. The chance is based on the external, towards the atomic game structure, procedures of constructing φ^I-strategy. We will call such φ^I-strategies *structural*. The well-known copy strategy is one of them. Obviously, if Opponent will play $\alpha\otimes\neg\alpha$ when the components of tensor are undetermined games, and if she/he copies moves of one game to another then it gives a φ^I-strategy, and it is a copy strategy.

Consequently, we found that the structural φ^I-strategies contain peculiarity of parallel operator towards the operators of choice. However, the appearance of structural φ^I-strategies for semantics of the parallel games causes delicacy in definition of the axiomatics. Semantics of game model is supported by the notion of an existence of φ^I-strategy for the game. Structural φ^I-strategies are components of φ^I-strategy; hence they provide an indirect influence on the game model. On the other hand, the structural φ^I-strategies are not completely determined. Nobody can protect against inventions of new structural φ^I-strategies inside the determined structures of parallel games. In this way, even if we will find a sound and complete axiomatics for the given structural φ^I-strategies, the invention of a new one can not guarantee that we will save the completeness of axiomatics.

4.2.5 Game model

In the basis we put the game semantics introduced by Pauly (1999) for Game Logic without parallelism. Given a *game model* $I = (S,\{G(a, s) \mid a\in \Pi_0 \text{ and } s\in S\}, V)$, where S is a set of states, $G(a, s)$ are *game structures* (determined dynamic games) on S, $V : \Phi_0 \to P(S)$ is the valuation function, define truth in a game model:

$I, s \vDash p$	iff	$p \in \Pi_0$ and $s \in V(p)$
$I, s \vDash \neg\varphi$	iff	$I, s \nvDash \varphi$
$I, s \vDash \varphi \vee \psi$	iff	$I, s \vDash \varphi$ or $I, s \vDash \psi$
$I, s \vDash [\pi]\varphi$	iff	Opponent has a φ^I- strategy in game $G(\pi, s)$
$I, s \vDash \langle\pi\rangle\varphi$	iff	Proponent has a φ^I- strategy in game $G(\pi, s)$

where $\varphi^I := \{s\in S \mid I, s \vDash \varphi\}$.

Game structure defined above can be included in the definition of game model as follows. In our notation $G(\pi, s)$ is a game π that is played from the given state s, and which is determined by correspondent game structure. States of the model are equal to the states of game π. We can establish the strict connection between valuation function $V : \Phi_0 \to P(S)$ and specified mask functions for Proponent $\mu^P: \boldsymbol{S} \to \{\top, \bot\}$ and another one for Opponent (depending on whose φ^I-strategy we would like to explore). For instance, suppose that player **N** has a winning strategy in game $G(\pi, s)$. If we will choose a valuation function $V : \varphi \to P(S)$ such that it is determined for at least all states that are outcome states of the winning strategy then the player **N** will have a φ^I-strategy in the game. Consequently, the valuation function can be connected with outcome states of the winning strategy. The states, in turn, depend on the specified mask functions.

4.3 Game operators for game structure

4.3.1 Non-parallel operators

Below we define some basic operators of Game Logic in terms of game structure determined in previous section.

Definition 4.10 ***Dual game***

Dual game $\gamma = \pi^d$ is a game structure, which can be obtained from the structure of game π after the switching of functions λ and μ between the players as follows:

$\forall s \in \boldsymbol{S}$: $(\lambda_\pi(s) = \mathbf{P} \leftrightarrow \lambda_\gamma(s) = \mathbf{O}) \wedge$

$(\lambda_\pi(s) = \mathbf{O} \leftrightarrow \lambda_\gamma(s) = \mathbf{P}) \wedge$

$(\lambda_\pi(s) = \mathbf{E} \leftrightarrow \lambda_\gamma(s) = \mathbf{E}) \wedge$

$(\mu^P{}_\pi(s) \leftrightarrow \mu^O{}_\gamma(s)) \wedge$

$(\mu^O{}_\pi(s) \leftrightarrow \mu^P{}_\gamma(s))$

Remark. ***Effect of $\langle\pi\rangle \leftrightarrow [\pi^d]$ on the expression of winning, φ^I-strategies.***

There is a fundamental game theoretic idea that if a player can succeed in a game, then her/his competitor can succeed in the dual game. Let us look at the effect of this idea on the expression of winning and φ^I-strategies, and game structure for non-parallel games.

1) If we talk in terms of a winning strategy, then in dual game objects of interests of Proponent and Opponent remain the same, namely their own win. Meanwhile the states, which correspond to these wins, are changed (the players are interchanging by the sets of their winning branches).

2) If we talk in terms of a φ^{I}-strategy, the object of interest of Proponent in a game becomes the

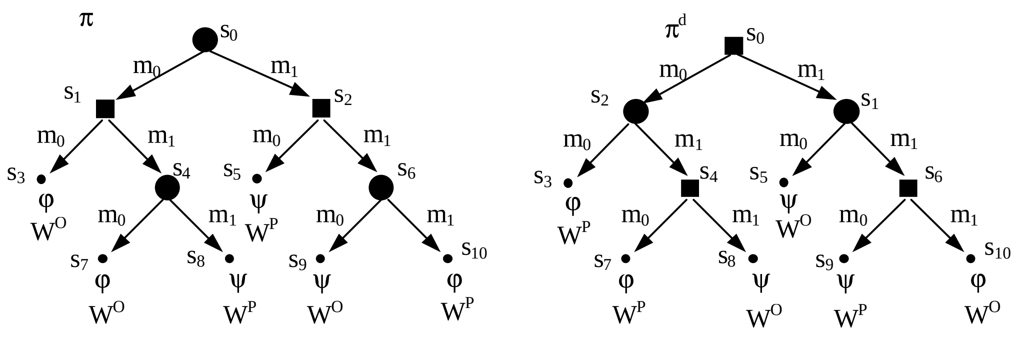


Figure 4.5 Changes in game structure for dual game.

object of interest of Opponent in a dual one (and vice versa). This means they are interested in achieving a state where φ is valid (or not valid). Simultaneously, the states where φ is valid is not changed.

From this it is easy to see that in both cases we have the same result, and it is the reason why we have a slightly different description of elements relating to winning and φ^{I}-strategies in game structure. In Figure 4.5 we depicted the situation of discussed changes in dual game. As you can see, φ^{I}-strategies may change from correspondence with winning strategy of a player to her/his loosing strategy even inside one model.

Definition 4.11 *Test game*[*]

Test game $\gamma = \varphi?$ is a game structure, which satisfies to the following properties.

1) $\boldsymbol{S}_\gamma = \{s_0\}$
2) $\boldsymbol{M}_\gamma = \{\langle\rangle\}$
3) $\boldsymbol{R}_\gamma = \{\langle\rangle\}$
4) $\chi_\gamma(\langle\rangle) = s_0$
5) $\lambda_\gamma(s_0) = \mathbf{E}$
6) $\mu^{P}{}_\gamma(s_0) = \perp$
7) $\mu^{O}{}_\gamma(s_0) = \top \leftrightarrow I, s_0 \nVdash \varphi$

Definition 4.12 *Union game*

Union game $\gamma = \alpha \cup \beta$ is a game structure, which satisfies to the following properties.

1) $\boldsymbol{S}_\gamma = \boldsymbol{S}_\alpha \cup \boldsymbol{S}_\beta \cup s_\cup$

[*] It will not be used in what follows

2) $\boldsymbol{M}_\gamma = \boldsymbol{M}_\alpha \cup \boldsymbol{M}_\beta \cup 0 \cup 1$

3) $\boldsymbol{R}_\gamma = \langle\rangle \cup \langle 0\rangle\boldsymbol{R}_\alpha \cup \langle 1\rangle\boldsymbol{R}_\beta$

4) $\forall \mathbf{p} \in \boldsymbol{R}_\alpha: \chi_\gamma(\langle 0\rangle\mathbf{p}) = \chi_\alpha(\mathbf{p}) \wedge$

$\forall \mathbf{p} \in \boldsymbol{R}_\beta: \chi_\gamma(\langle 1\rangle\mathbf{p}) = \chi_\beta(\mathbf{p}) \wedge$

$\chi_\gamma(\langle\rangle) = s_\cup$

5) $\forall s \in \boldsymbol{S}_\alpha: \lambda_\gamma(s) = \lambda_\alpha(s) \wedge$

$\forall s \in \boldsymbol{S}_\beta: \lambda_\gamma(s) = \lambda_\beta(s) \wedge$

$\lambda_\gamma(s_\cup) = \mathbf{P}$

6) $\forall s \in \boldsymbol{S}_\alpha: \mu^P{}_\gamma(s) = \mu^P{}_\alpha(s) \wedge$

$\forall s \in \boldsymbol{S}_\beta: \mu^P{}_\gamma(s) = \mu^P{}_\beta(s)$

7) $\forall s \in \boldsymbol{S}_\alpha: \mu^O{}_\gamma(s) = \mu^O{}_\alpha(s) \wedge$

$\forall s \in \boldsymbol{S}_\beta: \mu^O{}_\gamma(s) = \mu^O{}_\beta(s)$

Definition 4.13 ***Composition game***

Composition game $\gamma = \alpha;\beta$ is a game structure, which satisfies to the following properties.

1) $\boldsymbol{S}_\gamma = \boldsymbol{S}_\alpha \cup \boldsymbol{S}_\beta$ (the states of β has to be renamed for each new start from α)

2) $\boldsymbol{M}_\gamma = \boldsymbol{M}_\alpha \cup \boldsymbol{M}_\beta$ (the moves of β has to be renamed for each new start from α)

3) $\boldsymbol{R}_\gamma = \boldsymbol{R}_\alpha \cup \boldsymbol{R}_{\alpha^*} = \{\mathbf{p} \in \boldsymbol{R}_\alpha: (\lambda_\alpha(\chi_\alpha(\mathbf{p})) = \mathbf{E}) \wedge (\mu^P{}_\alpha(\chi_\alpha(\mathbf{p})) = \bot) \wedge (\mu^O{}_\alpha(\chi_\alpha(\mathbf{p})) = \bot)\}\boldsymbol{R}_\beta$

4) $\forall \mathbf{p} \in \boldsymbol{R}_\alpha - \boldsymbol{R}_{\alpha^*}: \chi_\gamma(\mathbf{p}) = \chi_\alpha(\mathbf{p}) \wedge$

$\forall \mathbf{p} \in \boldsymbol{R}_{\alpha^*}\boldsymbol{R}_\beta: ((\boldsymbol{R}_\beta \neq \varnothing) \rightarrow (\chi_\gamma(\mathbf{p}) = \chi_\beta(\mathbf{p}|_{\boldsymbol{R}_\beta}))) \wedge$

$((\boldsymbol{R}_\beta = \varnothing) \rightarrow (\chi_\gamma(\mathbf{p}) = \chi_\alpha(\mathbf{p})))$

5) $\forall s \in \boldsymbol{S}_\alpha: \lambda_\gamma(s) = \lambda_\alpha(s) \wedge$

$\forall s \in \boldsymbol{S}_\beta: \lambda_\gamma(s) = \lambda_\beta(s)$

6) $\forall s \in \boldsymbol{S}_\alpha: \mu^P{}_\gamma(s) = \mu^P{}_\alpha(s) \wedge$

$\forall s \in \boldsymbol{S}_\beta: \mu^P{}_\gamma(s) = \mu^P{}_\beta(s)$

7) $\forall s \in \boldsymbol{S}_\alpha: \mu^O{}_\gamma(s) = \mu^O{}_\alpha(s) \wedge$

$\forall s \in \boldsymbol{S}_\beta: \mu^O{}_\gamma(s) = \mu^O{}_\beta(s)$

4.3.2 Parallel operators

In this section we suppose to extend the existing language of Game Logic to parallel operators. We will introduce an analogue of multiplicative operators from Linear Logic, which were described in part 3. We have called new operators parallel; one of them is tensor, the other is par.

Analogue of Tensor game

Tensor game is a type of parallel game, in which the component games can be played alternatively and the switching between the games is governed by Opponent.

Definition 4.14 *Tensor game*

Tensor game $\gamma = \alpha \otimes \beta$ is a game structure, which satisfies the following properties.

1) $\boldsymbol{S}_\gamma = \{s = (s_\alpha, s_\beta) : s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta\}$ is a set of the *multiple*[*] states; multiple state is a *multiset*; If $\boldsymbol{S}_\alpha$ and $\boldsymbol{S}_\beta$ contain equal states, they have to be renamed.

2) $\boldsymbol{M}_\gamma = \boldsymbol{M}_\alpha \cup \boldsymbol{M}_\beta$; if α and β have equal names for some moves, then the moves have to be renamed (even if $\alpha = \beta$); we need it to know to what game each instantiation belongs to.

3) $\boldsymbol{R}_\gamma = \{\mathbf{q} : \mathbf{q}|_{\boldsymbol{M}_\alpha} \in \boldsymbol{R}_\alpha \ \wedge\ \mathbf{q}|_{\boldsymbol{M}_\beta} \in \boldsymbol{R}_\beta \ \wedge\ \forall\, \mathbf{p}x\, ((\mathbf{p}x \le \mathbf{q} \ \wedge\ x \in \boldsymbol{M}_\gamma) \rightarrow$

   $(\lambda_\gamma(\chi_\gamma(\mathbf{p})) = \lambda_\pi(\chi_\pi(\mathbf{p}|_{\boldsymbol{M}_\pi})) \ \leftrightarrow\ \mathbf{p}|_{\boldsymbol{M}_\pi} \ne \mathbf{p}x|_{\boldsymbol{M}_\pi}))$ for $\pi \in \{\alpha, \beta\}$

   (it means that the player who has to move at the current state of game γ can make moves only in that component α or β where she/he has current right to move; inequality at the right part of '$\leftrightarrow$' means that the move x is in component π, equality at the left part of '$\leftrightarrow$' means that the player who has to move next from the state $\chi_\gamma(\mathbf{p})$ of game γ can make a move in component π)

   ($q^i|_{\boldsymbol{M}_\beta}$ denote projection q^i that retains only moves $m \in \boldsymbol{M}_\beta$);

   note that in spite of multiple states, we do not propose multiple moves

4) $\chi_\gamma : \boldsymbol{R}_\gamma \rightarrow \boldsymbol{S}_\gamma$;

   $\forall\, \mathbf{q} \in \boldsymbol{R}_\gamma, s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta$: if $\chi_\alpha(\mathbf{q}|_{\boldsymbol{M}_\alpha}) = s_\alpha$ and $\chi_\beta(\mathbf{q}|_{\boldsymbol{M}_\beta}) = s_\beta$ then $\chi_\gamma(\mathbf{q}) = (s_\alpha, s_\beta)$

5) $\lambda_\gamma : \boldsymbol{S}_\gamma \rightarrow \{\mathbf{P}, \mathbf{O}\}$;

   $\forall\, s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta$:

   $$\begin{aligned}\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{O} \leftrightarrow & (\lambda_\alpha(s_\alpha) = \mathbf{O} \wedge \lambda_\beta(s_\beta) = \mathbf{O}) \vee \\ & (\lambda_\alpha(s_\alpha) = \mathbf{E} \wedge \lambda_\beta(s_\beta) = \mathbf{O}) \vee \\ & (\lambda_\alpha(s_\alpha) = \mathbf{O} \wedge \lambda_\beta(s_\beta) = \mathbf{E})\end{aligned}$$

   $\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{P} \leftrightarrow (\lambda_\alpha(s_\alpha) = \mathbf{P} \vee \lambda_\beta(s_\beta) = \mathbf{P})$

   $\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{E} \leftrightarrow (\lambda_\alpha(s_\alpha) = \mathbf{E} \wedge \lambda_\beta(s_\beta) = \mathbf{E})$

6) $\mu^P_\gamma : \boldsymbol{S}_\gamma \rightarrow \{\top, \bot\}$;

   $\forall\, s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta$:

   $\mu^P_\gamma(s_\alpha, s_\beta) = \mu^P_\alpha(s_\alpha) \wedge \mu^P_\beta(s_\beta)$

[*] We use the multiple states to save information about the active positions of all involved but abandoned games. The positions unambiguously allow returning to the abandoned games.

7) μ^O_γ: $\boldsymbol{S}_\gamma \rightarrow \{\top, \bot\}$;

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha,\, s_\beta \in \boldsymbol{S}_\beta$:

$\mu^O_\gamma(s_\alpha, s_\beta) = \mu^O_\alpha(s_\alpha) \vee \mu^O_\beta(s_\beta)$

Remark. ***Some properties of scheduling and specified mask functions for parallel operators.***

1) Scheduling function λ for parallel operators plays the role of switcher. That is, it determines where the players have to move. For instance, it specifies, in case of tensor, that if Proponent has to move in one of the component games, then she/he has to move in tensor game. According to this requirement Proponent can have an option to move in both components only at the initial segment of tensor game. At this moment she/he actually may choose which game to play, because only the scheduling function controls the switching between games. At the other moments of tensor game, when both players can move in component games according to the scheduling function, Proponent has to move in the tensor game. However, in such moments Proponent can play only in one of the components, and hence she/he will not have a choice where to play. After her/his playing, when the turns in both components will belong to

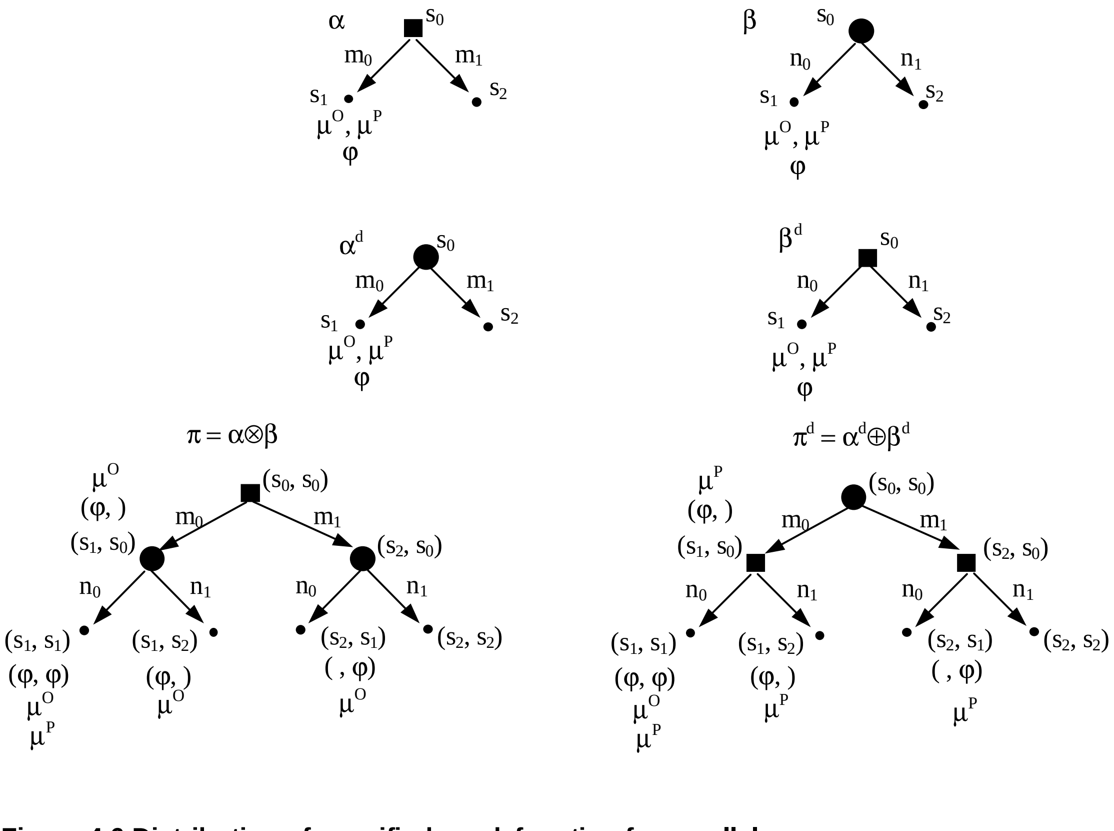


Figure 4.6 Distribution of specified mask function for parallel games.

Opponent, it is Opponent's turn to move in tensor game. In that situation Opponent will have the option to choose between two games, because in both of them it is her/his turn to move. Evidently the situation in which it is Proponent's turn to move in both components will never happen again.

It seems paradoxical that Opponent has more freedom for choice with scheduling function determined by conjunction (in tensor game) than Proponent with scheduling function determined by disjunction, because conjunction supposedly gives bigger restriction than disjunction!

2) When the component games of a parallel operator have a complex internal structure (including combinations of parallel operators), scheduling function λ also specifies where a player has to move. Thus, if the player can move in given component game and the component game has its own internal structure with parallel operators, then the player has to choose the main parallel operator of the component game and to apply the reasoning for the scheduling function to the level of that operator. If the player can not move in the component

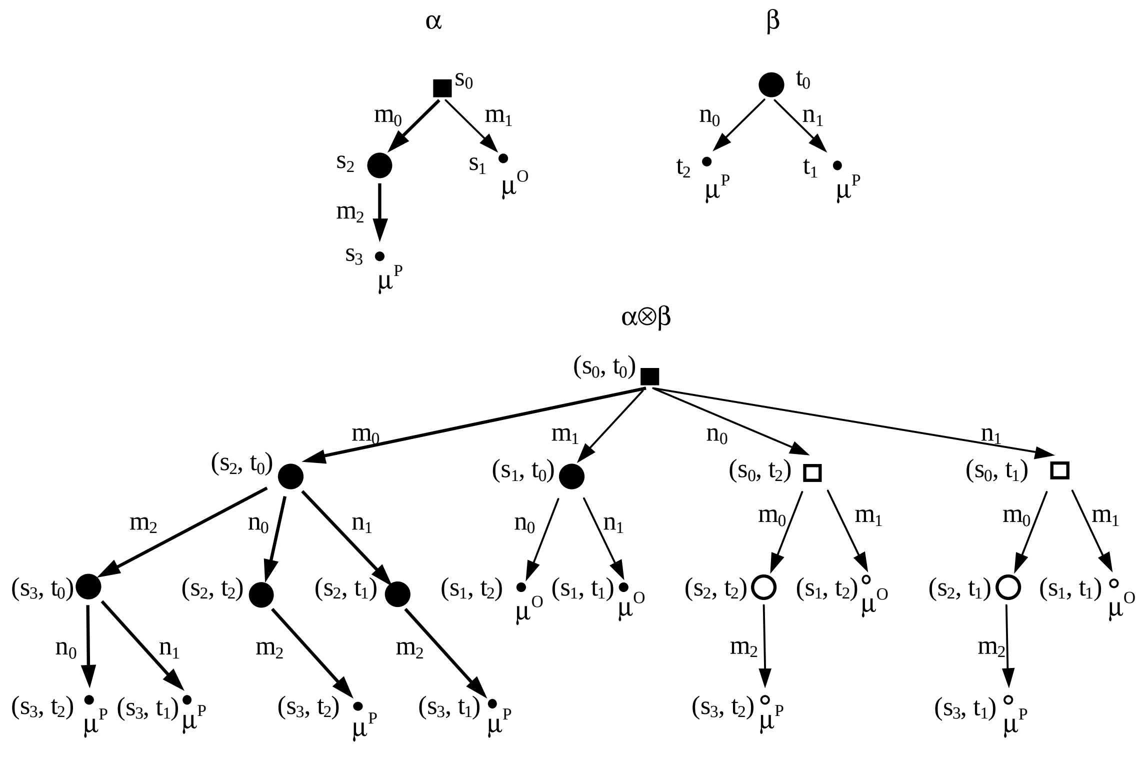


Figure 4.7 Game tree of a tensor game α⊗β obtained from two games α and β. We add multiple states to the elements depicted in Figure 4.1. Non-filled boxes, circles, and dots (together with substitution of Opponent's move on the place of initial node) correspond to the par game tree α⊕β.

game, then she/he can not move in any element of internal structure of that component even if she/he has turn to move there.

3) For atomic games specified mask function μ has to coincide for both players, otherwise it will mean that for one player the state validates φ whereas for the other this is not the case. Meanwhile, we need such an opportunity for the case of parallel operator. It is a reason why we determined this function. Parallel operators impose different requirements on the achievement of goals for each player. Thus tensor demands that Proponent could win both of the component games, whereas for Opponent it is enough to win only one of them. Par contains opposite requirements. So, instead of saying that μ corresponds to the states where φ is valid (it is true only for atomic games), it is more correct to say that μ is valid in the set of states which distribution in a game tree gives a task that the player has to complete in the considered game. The task of an atomic game for either player is to achieve a state where φ is valid, and it is a reason why μ coincides with such states. In case of tensor game the task for Proponent is to achieve a state where φ is valid in both components games, whereas for Opponent it is enough to find one such component. Hence domains μ^O and μ^P do not coincide in this case. In Figure 4.6 we depict an example of the distribution of specified mask function in different types of game.

Example of a tensor game tree

In Figure 4.2 we give an example of tree for tensor game. Game structure in this case has the following content ($\gamma = \alpha \otimes \beta$).

1) $\boldsymbol{S}_\gamma = \{(s_0, t_0), (s_2, t_0), (s_1, t_0), (s_0, t_2), (s_0, t_1), (s_3, t_0), (s_2, t_2), (s_2, t_1), (s_1, t_2), (s_1, t_1), (s_3, t_2), (s_3, t_1),\}$; when $\boldsymbol{S}_\alpha = \{s_0, s_1, s_2, s_3\}$; $\boldsymbol{S}_\beta = \{t_0, t_1, t_2\}$;
2) $\boldsymbol{M}_\gamma = \{m_0, m_1, m_2, n_0, n_1\}$
3) $\boldsymbol{R}_\gamma = \{\langle\ \rangle, \langle m_0\rangle, \langle m_1\rangle, \langle m_0, m_2\rangle, \langle m_0, n_0\rangle, \langle m_0, n_1\rangle, \langle m_1, n_0\rangle, \langle m_1, n_1\rangle, \langle m_0, n_0, m_2\rangle\}$
4) $\chi_\gamma(\langle\ \rangle) = (s_0, t_0)$; $\chi_\gamma(\langle m_0\rangle) = (s_2, t_0)$; $\chi_\gamma(\langle m_1\rangle) = (s_1, t_0)$ …
5) $\lambda_\gamma((s_0, t_0)) = \mathbf{P}$; $\lambda_\gamma((s_2, t_0)) = \mathbf{O}$; $\lambda_\gamma((s_1, t_0)) = \mathbf{O}$; $\lambda_\gamma((s_0, t_2)) = \mathbf{P}$; $\lambda_\gamma((s_3, t_2)) = \mathbf{E}$ …
6) $\mu^P{}_\gamma(s_3, t_1) = \top$; $\mu^P{}_\gamma(s_3, t_2) = \top$
7) $\mu^O{}_\gamma(s_1, t_0) = \top$; $\mu^O{}_\gamma(s_1, t_1) = \top$; $\mu^O{}_\gamma(s_1, t_2) = \top$

Analogue of Par game

Par game is a type of parallel game, in which the component games can be played alternatively and the switching between the games is governed by Proponent.

Definition 4.15 ***Par game***

Par game $\gamma = \alpha \oplus \beta$ is a game structure that is distinguished from the tensor game structure determined in definition 4.3 by the description of the following functions:

1) $\lambda_\gamma: \boldsymbol{S}_\gamma \rightarrow \{\mathbf{P}, \mathbf{O}\}$;

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta$:

$\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{O} \leftrightarrow (\lambda_\alpha(s_\alpha) = \mathbf{O} \vee \lambda_\beta(s_\beta) = \mathbf{O})$

$$\begin{aligned} \lambda_\gamma(s_\alpha, s_\beta) = \mathbf{P} \leftrightarrow\ & (\lambda_\alpha(s_\alpha) = \mathbf{P} \wedge \lambda_\beta(s_\beta) = \mathbf{P}) \vee \\ & (\lambda_\alpha(s_\alpha) = \mathbf{E} \wedge \lambda_\beta(s_\beta) = \mathbf{P}) \vee \\ & (\lambda_\alpha(s_\alpha) = \mathbf{P} \wedge \lambda_\beta(s_\beta) = \mathbf{E}) \end{aligned}$$

$\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{E} \leftrightarrow (\lambda_\alpha(s_\alpha) = \mathbf{E} \wedge \lambda_\beta(s_\beta) = \mathbf{E})$

2) $\mu^P_\gamma: \boldsymbol{S}_\gamma \rightarrow \{\top, \bot\}$;

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta$:

$\mu^P_\gamma(s_\alpha, s_\beta) = \mu^P_\alpha(s_\alpha) \vee \mu^P_\beta(s_\beta)$

3) $\mu^O_\gamma: \boldsymbol{S}_\gamma \rightarrow \{\top, \bot\}$;

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha, s_\beta \in \boldsymbol{S}_\beta$:

$\mu^O_\gamma(s_\alpha, s_\beta) = \mu^O_\alpha(s_\alpha) \wedge \mu^O_\beta(s_\beta)$

Further changes in game structure concern function $\boldsymbol{R}_\gamma = \boldsymbol{R}_\gamma(\lambda_\gamma)$.

Lemma 4.1 ***Stalemate states***

The game structures introduced in this section do not contain stalemate states, in which the game is not completed and a player who has to move can not do it because she/he has no admissible moves.

Proof

In an atomic game structure we defined the states, which do not contain moves as final. By applying parallel operators we do not obtain a stalemate state, because by definition of tensor and par at the final states of the game neither player has to move as in atomic game structure. Besides, it is easy to see that we can achieve the final states of both component games. Indeed, in parallel games a player has a move if she/he has a move in one of the component games, and hence she/he has something to do and deadlock is impossible.

$\times$

Lemma 4.2 ***Admissible branches***

The projections of the set of admissible branches of a parallel game on to the moves of component games contain all admissible branches of the component games.

Proof

By lemma 4.1 a parallel game does not contain stalemate states, i.e. a deadlock is impossible. Because we can finish both components by any branch, all of them have to be encapsulated in the set of admissible branches of the parallel game.

$\times$

4.3.3 Undeterminacy and lack of information

In Section 4.2.4 we discussed that the copy strategy has no sense in determined games. Here we introduce the axioms for copy strategy, considering lack of information as a reason of undeterminacy.

Copy strategy axioms:

$(s, s) \models [\pi \otimes \pi^d]\varphi$ (4.1)

$(s, s) \models \langle \pi \oplus \pi^d \rangle \varphi$ (4.2)

Theorem 4.1 ***Soundness of axioms for finite games with imperfect information***

Axioms (4.1), (4.2) are sound modulo φ-strategy (Definition 4.7) with respect to arbitrary interpretation of finite games with imperfect information and according to Definition 4.1, Definition 4.10, Definition 4.14, and Definition 4.15. For axioms (4.1), (4.2) we demand an extra condition, namely that the games $\pi \otimes \pi^d$ and $\pi \oplus \pi^d$ have to be φ^I-zero-sum.

Proof

(4.1) $(s, s) \models [\pi \otimes \pi^d]\varphi$

It is trivial when $\pi \otimes \pi^d$ is φ^I-zero-sum games. It means that by each branch either Proponent or Opponent can achieve a state where φ is valid and in all successor states (if their exists) φ is also valid. For Opponent in tensor game it is enough to achieve such a state only in one of the components π or π^d, and she/he can switch between them. Suppose Opponent will play alternatively in both components, copying moves from one to another (for that reason π and π^d. has to be played from the similar state). When the games are finished, Opponent occurs in such a multiple state (s_1, s_2) of game $\pi \otimes \pi^d$ that for state s_1: $\mu^P_\pi(s_1) = \top$ whereas for state s_2: $\mu^O_\pi(s_1) = \top$ (or vice versa), because s_1 and s_2 are equal states of dual φ^I-zero-sum games. Hence that is the definition of Opponent φ^I-strategy.

(4.2) Similar to the proof of (4.1)

$\times$

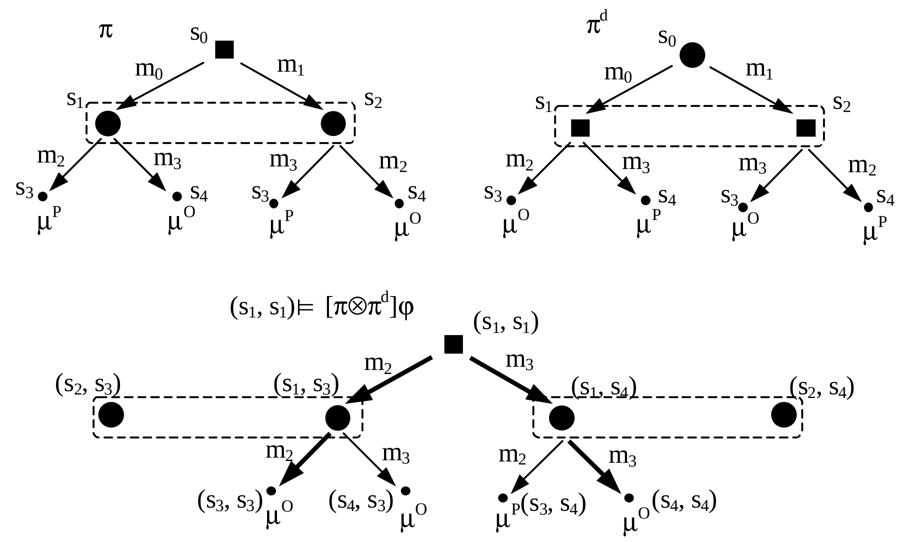


Figure 4.8 Example of copy strategy (stressed by bold arrows) of Opponent in tensor game structure $\pi\otimes\pi^d$. Here the states s_1 and s_2 are in the one information set for both Proponent and Opponent, i.e. they can not distinguish the states.

4.4 Axiomatics for parallel operators

In this section we present some axioms for games α, β, γ, π, and for arbitrary formula φ. The axioms for three types of games (infinite (two types), and finite determined) follow by proofs of soundness.

4.4.1 Axioms for infinite games

Arbitrary infinite games

Syntactic axioms:

$\models \langle\pi\rangle\varphi \leftrightarrow [\pi^d]\varphi$ (4.3)

$\models \langle\pi\rangle\varphi \leftrightarrow \langle(\pi^d)^d\rangle\varphi$ (4.4)

$\models \langle\pi\rangle\varphi \rightarrow \neg[\pi]\neg\varphi$ (4.5)

Symmetry and associative axioms for tensor and par:

$\models \langle\alpha\otimes\beta\rangle\varphi \leftrightarrow \langle\beta\otimes\alpha\rangle\varphi$ (4.6) $\models [\alpha\otimes\beta]\varphi \leftrightarrow [\beta\otimes\alpha]\varphi$ (4.7)

$\models \langle\alpha\oplus\beta\rangle\varphi \leftrightarrow \langle\beta\oplus\alpha\rangle\varphi$ (4.8) $\models [\alpha\oplus\beta]\varphi \leftrightarrow [\beta\oplus\alpha]\varphi$ (4.9)

$\models \langle\alpha\otimes(\beta\otimes\gamma)\rangle\varphi \leftrightarrow \langle(\alpha\otimes\beta)\otimes\gamma\rangle\varphi$ (4.10) $\models [\alpha\otimes(\beta\otimes\gamma)]\varphi \leftrightarrow [(\alpha\otimes\beta)\otimes\gamma]\varphi$ (4.11)

$\vDash \langle\alpha\oplus(\beta\oplus\gamma)\rangle\varphi \leftrightarrow \langle(\alpha\oplus\beta)\oplus\gamma\rangle\varphi$ (4.12) $\vDash [\alpha\oplus(\beta\oplus\gamma)]\varphi \leftrightarrow [(\alpha\oplus\beta)\oplus\gamma]\varphi$ (4.13)

Axioms of mutation:

$\vDash \langle(\alpha\otimes\beta)^d\rangle\varphi \leftrightarrow \langle\alpha^d\oplus\beta^d\rangle\varphi$ (4.14)

$\vDash [(\alpha\oplus\beta)^d]\varphi \leftrightarrow [\alpha^d\otimes\beta^d]\varphi$ (4.15)

Parallel operators' reduction axioms:

$\vDash \langle\alpha\otimes\beta\rangle\varphi \rightarrow \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\varphi$ (4.16)

$\vDash [\alpha\oplus\beta]\varphi \rightarrow [\alpha]\varphi \wedge [\beta]\varphi$ (4.17)

Theorem 4.2 *Soundness of axioms for arbitrary infinite games*

Axioms (4.3) – (4.17) are sound modulo φ-strategy (Definition 4.7) with respect to arbitrary interpretation of infinite games and according with Definition 4.1, Definition 4.10, Definition 4.14, and Definition 4.15.

Proof

(4.3) $\vDash \langle\pi\rangle\varphi \leftrightarrow [\pi^d]\varphi$

$\rightarrow$: Assume Proponent has a φ^I-strategy ***U*** in game π.

We define a strategy ***T*** of dual game π^d as ***T*** = ***U***.

If ***U*** is a φ^I-strategy of Proponent in game π, then ***T*** is a φ^I-strategy of Opponent in game π^d.

Indeed, dual game contains the same set of admissible branches. In addition, the turns of Proponent in game π belong to Opponent in game π^d. Also, in the states where specified mask function of Proponent is valid in game π, the specified mask function of Opponent is valid in game π^d. Hence, Opponent will have a φ^I-strategy in π^d and by construction it is ***T***.

$\leftarrow$: The backward proof is similar.

(4.4) $\vDash \langle\pi\rangle\varphi \leftrightarrow \langle(\pi^d)^d\rangle\varphi$

It obviously follows from Definition 4.10 of dual game, because $\pi \equiv (\pi^d)^d$.

(4.5) $\vDash \langle\pi\rangle\varphi \rightarrow \neg[\pi]\neg\varphi$

The states where φ is valid, ¬φ is not valid (and vice versa). Since Proponent has a φ^I-strategy, she/he can reach a state where φ is valid and for all successor states (if they exists) φ is also valid. Hence Opponent can not do the same for function ¬φ.

(4.6) $\vDash \langle\alpha\otimes\beta\rangle\varphi \leftrightarrow \langle\beta\otimes\alpha\rangle\varphi$

By definition of tensor game, the game-structures for $\alpha\otimes\beta$ and $\beta\otimes\alpha$ are equal.

Note that Proponent has a φ^I-strategies in tensor games $\alpha\otimes\beta$ and $\beta\otimes\alpha$ if both of the games α and β are finite. Indeed, if one of them is infinite, then Opponent can keep playing in that game and hence in the other game Proponent will never reach a state where φ is valid (the same for successor states). Therefore Proponent will never complete a demand for φ^I-strategy in tensor game $\alpha\otimes\beta$ for her/him. But it does not confuse equivalence.

(4.7) $\vDash [\alpha\otimes\beta]\varphi \leftrightarrow [\beta\otimes\alpha]\varphi$

By definition of tensor game, the game-structures for $\alpha\otimes\beta$ and $\beta\otimes\alpha$ are equal. In case of infinite initial segment, Opponent has a φ^I-strategy in the tensor games if and only if it her/his turn to move in the segment. Otherwise Proponent will keep playing in the segment which can prevent Opponent to reach a state where φ valid and for all successor states (if their exists) φ is also valid. But it does not confuse equivalence.

(4.9), (4.10), (4.13) Similar to the proof of (4.6).

(4.8), (4.11), (4.12) Similar to the proof of (4.7).

(4.14) $\vDash \langle(\alpha\otimes\beta)^d\rangle\varphi \leftrightarrow \langle\alpha^d\oplus\beta^d\rangle\varphi$

By definitions of dual, tensor, and par games, the game-structures for $(\alpha\otimes\beta)^d$ and $\alpha^d\oplus\beta^d$ are equal. Since the equality for the elements of game structures $(\alpha\otimes\beta)^d$ and $\alpha^d\oplus\beta^d$ is not so obvious as in case of axiom (4.5), except sets of states and moves, we will show this equality in detail.

Let $\pi = \alpha\otimes\beta$, $\gamma = \pi^d$, $\delta = \alpha^d$, $\varepsilon = \beta^d$, $\sigma = \delta \oplus \varepsilon$.

1) We prove that $\lambda_\gamma(s_\alpha, s_\beta) = \lambda_\sigma(s_\alpha, s_\beta)$.

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha,\ s_\beta \in \boldsymbol{S}_\beta$:

Show that $\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{P} \leftrightarrow \lambda_\sigma(s_\alpha, s_\beta) = \mathbf{P}$:

$\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{P} \leftrightarrow$

$\lambda_\pi(s_\alpha, s_\beta) = \mathbf{O} \leftrightarrow$

$(\lambda_\alpha(s_\alpha) = \mathbf{O} \wedge \lambda_\beta(s_\beta) = \mathbf{O}) \vee (\lambda_\alpha(s_\alpha) = \mathbf{E} \wedge \lambda_\beta(s_\beta) = \mathbf{O}) \vee (\lambda_\alpha(s_\alpha) = \mathbf{O} \wedge \lambda_\beta(s_\beta) = \mathbf{E}) \leftrightarrow$

$(\lambda_\delta(s_\alpha) = \mathbf{P} \wedge \lambda_\varepsilon(s_\beta) = \mathbf{P}) \vee (\lambda_\delta(s_\alpha) = \mathbf{E} \wedge \lambda_\varepsilon(s_\beta) = \mathbf{P}) \vee (\lambda_\delta(s_\alpha) = \mathbf{P} \wedge \lambda_\varepsilon(s_\beta) = \mathbf{E}) \leftrightarrow$

$\lambda_\sigma(s_\alpha, s_\beta) = \mathbf{P}$

Show that $\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{O} \leftrightarrow \lambda_\sigma(s_\alpha, s_\beta) = \mathbf{O}$:

$\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{O} \leftrightarrow$

$\lambda_\pi(s_\alpha, s_\beta) = \mathbf{P} \leftrightarrow$

$(\lambda_\alpha(s_\alpha) = \mathbf{P} \vee \lambda_\beta(s_\beta) = \mathbf{P}) \leftrightarrow$

$(\lambda_\delta(s_\alpha) = \mathbf{O} \vee \lambda_\varepsilon(s_\beta) = \mathbf{O}) \leftrightarrow$

$\lambda_\sigma(s_\alpha, s_\beta) = \mathbf{O}$

Show that $\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{E} \leftrightarrow \lambda_\sigma(s_\alpha, s_\beta) = \mathbf{E}$:

$\lambda_\gamma(s_\alpha, s_\beta) = \mathbf{E} \leftrightarrow$

$\lambda_\pi(s_\alpha, s_\beta) = \mathbf{E} \leftrightarrow$

$(\lambda_\alpha(s_\alpha) = \mathbf{E} \wedge \lambda_\beta(s_\beta) = \mathbf{E}) \leftrightarrow$

$(\lambda_\delta(s_\alpha) = \mathbf{E} \wedge \lambda_\varepsilon(s_\beta) = \mathbf{E}) \leftrightarrow$

$\lambda_\sigma(s_\alpha, s_\beta) = \mathbf{E}$

2) We prove that $\mu^O{}_\gamma(s_\alpha, s_\beta) = \mu^O{}_\sigma(s_\alpha, s_\beta)$:

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha,\ s_\beta \in \boldsymbol{S}_\beta$:

$\mu^O{}_\gamma(s_\alpha, s_\beta) = \mu^P{}_\pi(s_\alpha, s_\beta) = \mu^P{}_\alpha(s_\alpha) \wedge \mu^P{}_\beta(s_\beta) = \mu^O{}_\delta(s_\alpha) \wedge \mu^O{}_\varepsilon(s_\beta) = \mu^O{}_\sigma(s_\alpha, s_\beta)$

3) We prove that $\mu^P{}_\gamma(s_\alpha, s_\beta) = \mu^P{}_\sigma(s_\alpha, s_\beta)$:

$\forall\, s_\alpha \in \boldsymbol{S}_\alpha,\ s_\beta \in \boldsymbol{S}_\beta$:

$\mu^P{}_\gamma(s_\alpha, s_\beta) = \mu^O{}_\pi(s_\alpha, s_\beta) = \mu^O{}_\alpha(s_\alpha) \vee \mu^O{}_\beta(s_\beta) = \mu^P{}_\delta(s_\alpha) \vee \mu^P{}_\varepsilon(s_\beta) = \mu^P{}_\sigma(s_\alpha, s_\beta)$

4) We prove that $\boldsymbol{R}_\gamma = \boldsymbol{R}_\sigma$ (note that $\boldsymbol{R}_{\alpha\otimes\beta} \neq \boldsymbol{R}_{\alpha\oplus\beta}$, see Figure 4.7):

$\boldsymbol{R}_\gamma = \boldsymbol{R}_\pi = \{\mathbf{q}\colon \mathbf{q}|_{\boldsymbol{M}_\alpha} \in \boldsymbol{R}_\alpha \;\wedge\; \mathbf{q}|_{\boldsymbol{M}_\beta} \in \boldsymbol{R}_\beta \;\wedge\; \forall\, \mathbf{p}\boldsymbol{x}\ ((\mathbf{p}\boldsymbol{x} \le \mathbf{q} \;\wedge\; \boldsymbol{x} \in \boldsymbol{M}_\pi) \rightarrow$

$(\lambda_\pi(\chi_\pi(\mathbf{p})) = \lambda_\eta(\chi_\eta(\mathbf{p}|_{\boldsymbol{M}_\eta})) \;\leftrightarrow\; \mathbf{p}|_{\boldsymbol{M}_\eta} \neq \mathbf{p}\boldsymbol{x}|_{\boldsymbol{M}_\eta}))$ for $\eta \in \{\alpha, \beta\}$

$\boldsymbol{R}_\sigma = \{\mathbf{q}\colon \mathbf{q}|_{\boldsymbol{M}_\delta} \in \boldsymbol{R}_\delta \;\wedge\; \mathbf{q}|_{\boldsymbol{M}_\varepsilon} \in \boldsymbol{R}_\varepsilon \;\wedge\; \forall\, \mathbf{p}\boldsymbol{x}\ ((\mathbf{p}\boldsymbol{x} \le \mathbf{q} \;\wedge\; \boldsymbol{x} \in \boldsymbol{M}_\sigma) \rightarrow$

$(\lambda_\sigma(\chi_\sigma(\mathbf{p})) = \lambda_\nu(\chi_\nu(\mathbf{p}|_{\boldsymbol{M}_\nu})) \;\leftrightarrow\; \mathbf{p}|_{\boldsymbol{M}_\nu} \neq \mathbf{p}\boldsymbol{x}|_{\boldsymbol{M}_\nu}))$ for $\nu \in \{\delta, \varepsilon\}$

Obviously, by definition of dual game, it follows that

$\boldsymbol{R}_\alpha = \boldsymbol{R}_\delta$, $\boldsymbol{R}_\beta = \boldsymbol{R}_\varepsilon$, $\boldsymbol{M}_\alpha = \boldsymbol{M}_\delta$, $\boldsymbol{M}_\beta = \boldsymbol{M}_\varepsilon$,

and by definitions of dual, tensor and par games it follows that $\boldsymbol{M}_\pi = \boldsymbol{M}_\sigma$.

Thus, to show that $\boldsymbol{R}_\gamma = \boldsymbol{R}_\sigma$, it remains to show that

$\forall\, \mathbf{p}\ (\lambda_\pi(\chi_\pi(\mathbf{p})) = \lambda_\eta(\chi_\eta(\mathbf{p}|_{\boldsymbol{M}_\eta})) \leftrightarrow (\lambda_\sigma(\chi_\sigma(\mathbf{p})) = \lambda_\nu(\chi_\nu(\mathbf{p}|_{\boldsymbol{M}_\nu}))$ for $\eta \in \{\alpha, \beta\}$, $\nu \in \{\delta, \varepsilon\}$

From clause (1) of the proof:

$(\lambda_\pi = \mathbf{P}) \leftrightarrow (\lambda_\sigma = \mathbf{O}) \wedge (\lambda_\pi = \mathbf{O}) \leftrightarrow (\lambda_\sigma = \mathbf{P}) \wedge (\lambda_\pi = \mathbf{E}) \leftrightarrow (\lambda_\sigma = \mathbf{E}) \wedge$

$(\lambda_\alpha = \mathbf{P}) \leftrightarrow (\lambda_\delta = \mathbf{O}) \wedge (\lambda_\alpha = \mathbf{O}) \leftrightarrow (\lambda_\delta = \mathbf{P}) \wedge (\lambda_\alpha = \mathbf{E}) \leftrightarrow (\lambda_\delta = \mathbf{E}) \wedge$

$(\lambda_\beta = \mathbf{P}) \leftrightarrow (\lambda_\varepsilon = \mathbf{O}) \wedge (\lambda_\beta = \mathbf{O}) \leftrightarrow (\lambda_\varepsilon = \mathbf{P}) \wedge (\lambda_\beta = \mathbf{E}) \leftrightarrow (\lambda_\varepsilon = \mathbf{E})$

Hence, $(\lambda_\pi = \lambda_\eta) \leftrightarrow (\lambda_\sigma = \lambda_\nu)$ such that $((\eta = \alpha) \leftrightarrow (\nu = \delta)) \wedge ((\eta = \beta) \leftrightarrow (\nu = \varepsilon))$

(4.15) Similar to the proof of (4.14)

(4.16) $\vDash \langle\alpha\otimes\beta\rangle\varphi \rightarrow \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\varphi$

Let Proponent has a φ^I-strategy $\boldsymbol{T}$ in tensor game $\alpha\otimes\beta$.

Define a strategy $\boldsymbol{U}$ of game α as a projection of the φ^I-strategy $\boldsymbol{T}$, which contains only moves of game α: $\boldsymbol{U} = \boldsymbol{T}|_{\boldsymbol{M}_\alpha}$. Also define a strategy $\boldsymbol{V}$ of game β as a projection of the φ^I-strategy $\boldsymbol{T}$, which contains only moves of game β: $\boldsymbol{V} = \boldsymbol{T}|_{\boldsymbol{M}_\beta}$.

If $\boldsymbol{T}$ is a φ^I-strategy of Proponent in game $\alpha\otimes\beta$, then $\boldsymbol{U}$ is a φ^I-strategy of Proponent in game α, and $\boldsymbol{V}$ is a φ^I-strategy of Proponent $\boldsymbol{V}$ in game β.

Indeed, the fact that Proponent has a φ^I-strategy in tensor game $\alpha\otimes\beta$ means that she/he can achieve independently (because the switch of tensor controls Opponent) α and β states in both games, where φ is valid and in all successors (if they exists) φ is also valid. Hence Proponent has φ^I-strategies in both α and β, and by construction they are ***U*** and ***V*** respectively. ***U*** belongs to the set of admissible branches of game α and ***V*** belongs to the set of admissible branches of game β, because they were built by projection from ***T***. In turn ***T*** belongs to the set of admissible branches of game $\alpha\otimes\beta$ which projection by definition of tensor game belongs to the sets of admissible branches of games α and β.

Note that Proponent has a φ^I-strategy in tensor game $\alpha\otimes\beta$ if both of the games α and β are finite. Indeed, if one of them is infinite, then Opponent can keep playing in that game and hence in the other game Proponent will never reach a state where φ is valid (the same for successor states). Therefore Proponent will never complete a demand for φ^I-strategy in tensor game $\alpha\otimes\beta$ for her/him.

(4.17) Similar to the proof of (4.16)

$\times$

Infinite games with finite initial segment

Parallel operators' reduction axioms.

$\vDash [\alpha]\varphi \vee [\beta]\varphi \to [\alpha\otimes\beta]\varphi$ (4.18)

$\vDash \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi \to \langle\alpha\oplus\beta\rangle\varphi$ (4.19)

Theorem 4.3 ***Soundness of axioms for infinite games with finite initial segment***

Axioms (4.3) – (4.19) are sound modulo φ-strategy (Definition 4.7) with respect to arbitrary interpretation of infinite games with finite initial segment and according to Definition 4.1, Definition 4.10, Definition 4.14, and Definition 4.15.

Proof

(4.18) $\vDash [\alpha]\varphi \vee [\beta]\varphi \to [\alpha\otimes\beta]\varphi$

$\to$: Assume Opponent has a φ^I-strategy ***U*** in game α or a φ^I-strategy ***V*** in game β.

We define a strategy ***T*** of tensor game $\alpha\otimes\beta$ as ***T*** = ***U*** if Opponent has a φ^I-strategy in α, otherwise as ***T*** = ***V***.

If ***U*** is a φ^I-strategy of Opponent in game α or ***V*** is a φ^I-strategy of Opponent ***V*** in game β, then ***T*** is a φ^I-strategy of Opponent in game $\alpha\otimes\beta$.

Indeed, because the Opponent has the choice of which game to play in tensor game, she/he can follow according to where she/he has a φ^I-strategy by the moves of the φ^I-strategy of that game (α or β). This allows Opponent to achieve a state where φ is valid in one of the games. According to definition of tensor game it is enough for her/him to have a φ^I-strategy in tensor game $\alpha\otimes\beta$, and by construction it is $\boldsymbol{T}$. By lemma 4.2 since $\boldsymbol{U}$ and $\boldsymbol{V}$ belong to the sets of admissible branches of component games α and β, $\boldsymbol{T}$ has to belong to the set of admissible branches of tensor game.

The axiom may not be valid if an initial segment of the game where Opponent does not have φ^{I}-strategy is infinite, and it is Proponent's turn to move there, hence in the tensor game also. Therefore Opponent will never reach a state where φ is valid and in all successor states (if their exists) φ is also valid, and it means that she/he does not have a φ^{I}-strategy in this case.

(4.19) Similar to the proof of (4.18)

$\times$

4.4.2 Axioms for finite determined games

Syntactic axiom:

$\vDash \langle\pi\rangle\varphi \leftrightarrow \neg[\pi]\neg\varphi$ (4.20)

Parallel operators' reduction axioms:

$\vDash \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\varphi \leftrightarrow \langle\alpha\otimes\beta\rangle\varphi$ (4.21)

$\vDash [\alpha]\varphi \wedge [\beta]\varphi \leftrightarrow [\alpha\oplus\beta]\varphi$ (4.22)

Deterministic axioms:

$\vDash \langle\pi\rangle\varphi \vee \langle\pi^{d}\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^{d}\rangle\varphi$ (4.23)

$\vDash [\pi]\varphi \vee [\pi^{d}]\neg\varphi \vee [\pi]\neg\varphi \vee [\pi^{d}]\varphi$ (4.24)

Deterministic axiom is stronger than for the copy strategy:

$\vDash (\langle\pi\rangle\varphi \vee \langle\pi^{d}\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^{d}\rangle\varphi)\rightarrow((\langle\pi\oplus\pi^{d}\rangle\varphi \vee \langle\pi\oplus\pi^{d}\rangle\neg\varphi)\wedge([\pi\otimes\pi^{d}]\varphi \vee [\pi\otimes\pi^{d}]\neg\varphi))$ (4.25)

Deterministic axiom changes the axioms (4.18) and (4.19) ***(in this case we have no distinction between operators of choice and parallel operators!)***:

$\vDash (\langle\pi\rangle\varphi \vee \langle\pi^{d}\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^{d}\rangle\varphi)\rightarrow(([\alpha]\varphi \vee [\beta]\varphi \leftrightarrow [\alpha\otimes\beta]\varphi)\wedge(\langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi \leftrightarrow \langle\alpha\oplus\beta\rangle\varphi))$ (4.26)

Theorem 4.4 ***Soundness of axioms for finite determined games***

Axioms (4.3) – (4.26) are sound modulo φ-strategy (Definition 4.7) with respect to arbitrary interpretation of finite determined games and according to Definition 4.1, Definition 4.10, Definition 4.14, and Definition 4.15.

Proof

(4.20) $\vDash \langle\pi\rangle\varphi \leftrightarrow \neg[\pi]\neg\varphi$

$\rightarrow$: According to the axiom (4.5)

$\leftarrow$: If the game is determined and finite then the players can have a shadow of strategy couple according to Claim 4.1. Furthermore, if Opponent does not have ψ^I-strategy (where $\psi = \neg\varphi$) then the shadow of strategy couple can be either $(\top, \top, \bot, \bot)$ or $(\top, \bot, \top, \bot)$, and hence Proponent would have a φ^I-strategy.

(4.21) $\vDash \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\varphi \leftrightarrow \langle\alpha\otimes\beta\rangle\varphi$

$\rightarrow$: Assume Proponent has a φ^I-strategy ***U*** in game α and a φ^I-strategy ***V*** in game β.

We define a strategy ***T*** = ***T***(***U***, ***V***) of game $\alpha\otimes\beta$ as a set of sequences of moves which is constructed according to the clauses (3) and (5) of tensor structure applied to admissible branches ***U*** and ***V*** of games α and β.

If ***U*** is a φ^I-strategy of Proponent in game α, and ***V*** is a φ^I-strategy of Proponent ***V*** in game β, then ***T*** is a φ^I-strategy of Proponent in game $\alpha\otimes\beta$.

Indeed, although Proponent has no choice which game to play in tensor game, she/he can follow where she/he has to play by the moves of φ^I-strategy of that game. Since Proponent has φ^I-strategy in both α and β, and it follows from lemma 4.1 that deadlock is impossible, and since the games are finite, she/he can achieve states where φ is valid in both these games. This is enough for Proponent to have φ^I-strategy in tensor game $\alpha\otimes\beta$ and by construction it is ***T***.

The axiom may not be valid if either of game α or β is infinite. Since Opponent can switch the tensor she/he can keep playing in that game and hence in the other game Proponent will never play and will never reach a state where φ is valid (the same for successor states). Therefore Proponent will never complete a demand for her/his φ^I-strategy in tensor game $\alpha\otimes\beta$.

$\leftarrow$: By axiom (4.16)

(4.22) Similar to the proof of (4.21)

(4.23) $\vDash \langle\pi\rangle\varphi \vee \langle\pi^d\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^d\rangle\varphi$

Obviously follows from the Claim 4.1 for finite games.

(4.24) Similar to the proof of (4.23)

(4.25) $\vDash (\langle\pi\rangle\varphi \vee \langle\pi^d\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^d\rangle\varphi)\rightarrow(\langle\pi\oplus\pi^d\rangle\varphi \wedge [\pi\otimes\pi^d]\varphi)$

(i) $(\langle\pi\rangle\varphi \vee \langle\pi^d\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^d\rangle\varphi)\leftrightarrow([\pi]\varphi \vee [\pi^d]\neg\varphi \vee [\pi]\neg\varphi \vee [\pi^d]\varphi)$ by (4.3)

(ii) $(\langle\pi\rangle\varphi \vee \langle\pi^d\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^d\rangle\varphi)\rightarrow\langle\pi\oplus\pi^d\rangle\varphi$ by (4.19)

(iii) $([\pi]\varphi \vee [\pi^d]\neg\varphi \vee [\pi]\neg\varphi \vee [\pi^d]\varphi)\rightarrow[\pi\otimes\pi^d]\varphi$ by (4.18)

(iv) $(\langle\pi\rangle\varphi \vee \langle\pi^d\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^d\rangle\varphi)\rightarrow(\langle\pi\oplus\pi^d\rangle\varphi \wedge [\pi\otimes\pi^d]\varphi)$ from (i), (ii), (iii)

(4.26) $\vDash (\langle\pi\rangle\varphi \vee \langle\pi^d\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^d\rangle\varphi)\rightarrow(([\alpha]\varphi \vee [\beta]\varphi \leftrightarrow [\alpha\otimes\beta]\varphi)\wedge(\langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi \leftrightarrow \langle\alpha\oplus\beta\rangle\varphi))$

We will prove $(\langle\pi\rangle\varphi \vee \langle\pi^{d}\rangle\neg\varphi \vee \langle\pi\rangle\neg\varphi \vee \langle\pi^{d}\rangle\varphi)\rightarrow(\langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi \leftrightarrow \langle\alpha\oplus\beta\rangle\varphi)$, the rest of the proof has an idea similar to the proof of axiom (4.25).

$\langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi \rightarrow \langle\alpha\oplus\beta\rangle\varphi$ is an axiom (4.19).

Consider $\langle\alpha\oplus\beta\rangle\varphi \rightarrow \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi$.

Suppose Proponent has a φ^{I}-strategy in $\alpha\oplus\beta$ and has none in either α or β. Then according to Claim 4.1 Opponent has ψ^{I}-strategy (where $\psi = \neg\varphi$) in both α and β. Hence by (4.22) $[\alpha]\varphi \wedge [\beta]\varphi \leftrightarrow [\alpha\oplus\beta]\varphi$. We reach a contradiction: two-player can never have φ^{I} and ψ^{I} strategies simultaneously in one game, including $\alpha\oplus\beta$.

×

5 Discussion

5.1 Structural strategies for parallel games

In the previous chapter we determined only one *structural* strategy, namely copy strategy for parallel operators. However, Linear Logic contains operators of repetition which we have not considered. The operators allow the determination of other copy strategies. It would spread out the semantics of Game Logic if we had taken it into account in the previous chapter. It is not even clear how many structural strategies exist for given operator. Besides, in this section we offer an structural φ^I-strategy for operators of repetition, which we also determine below. It is possible to extend Game Logic with operators of repetition, but we only sketch the idea.

5.1.1 Operators of repetition

In this subsection we introduce the operators of repetition, which contains in Linear Logic.

Definition 5.1 ***Repetition "of course"***[*]

$\boldsymbol{R(A)} = (\boldsymbol{M_{R(A)}} , \boldsymbol{\lambda_{R(A)}}, \boldsymbol{P_{R(A)}} , \boldsymbol{W_{R(A)}})$

Where:

- $\boldsymbol{M_{R(A)}}$ is the partial Cartesian product of the possible moves' set $\boldsymbol{M_A}$: $\boldsymbol{M_A} \times\ldots\times \boldsymbol{M_A}$, as it shown in Figure 5.1. It is constructed as follows.
  1) Take the initial state of game $\boldsymbol{A}$ as an initial state of the game with repetition.
  2) The other states of the game with repetition can be in complex form, obtained by adding extra-opened play of game $\boldsymbol{A}$.
  3) A state of a game can have the following successors:
     - states, whose components are equal to components of the predecessor state, except one, that is substituted by successor state in initial game
     - state, whose components are equal to components of the predecessor state, but which has one extra component, that is equal to the initial state of the initial game.
- $\boldsymbol{\lambda_{R(A)}}$ labelling function: It is Opponent's to move if and only if it is his turn to move in all opened games, otherwise it is Proponent's turn to move (note that if Proponent would have a first move in the initial game then Opponent will not have a winning strategy, because Proponent can continuously open a new game)

[*] By Girard's (1987) notation

A

$$\begin{tikzcd}
 & \alpha_0 \arrow[dl] \arrow[dr] & \\
\alpha_1 \arrow[d] & & \alpha_2 \\
\alpha_3 & &
\end{tikzcd}$$

R(A) or $\bar{R}$ *(A)*

$$\begin{tikzcd}
 & \alpha_0^1 \arrow[dl] \arrow[dr] \arrow[rrr] & & & (\alpha_0^1,\alpha_0^2) \arrow[dl] \arrow[d] \arrow[dr] \arrow[rr] & & (\alpha_0^1,\alpha_0^2,\alpha_0^3) \\
\alpha_1^1 \arrow[d] \arrow[dr] & & \alpha_2^1 & (\alpha_1^1,\alpha_0^2) & (\alpha_1^1,\alpha_0^2) & (\alpha_1^1,\alpha_0^2) & \\
\alpha_3^1 & (\alpha_1^1,\alpha_0^2) \arrow[dl] \arrow[d] \arrow[dr] \arrow[rr] & & (\alpha_1^1,\alpha_0^2,\alpha_0^3) & & & \\
(\alpha_3^1,\alpha_0^2) & (\alpha_1^1,\alpha_1^2) & (\alpha_1^1,\alpha_2^2) & & & &
\end{tikzcd}$$

Figure 5. 1. The set of all possible moves in compound game with a multiplicative repetition is the following partial Cartesian product of the initial game (superscript is the number of the opened game, subscript is the number of move in the game)

- ***P*** $_{R(A)}$ is the set of valid positions (admissible moves) that is constructed as follows:
  1) If it is Opponent's turn to move then he can move in any possible direction (during his turn he is allowed to move in all of the components, and hence he can always switch the game)
  2) If it is Proponent's turn to move then she can move only in the components where it is her turn to move (note that if it is her turn to move in several components then she can switch the games and open a new game)
- ***W*** $_{R(A)}$ is the set of all winning for Proponent branches that equal to intersection of the winning branches of the initial game (i.e. to win the game, Proponent has to win in all of the opened components of the resulting repetitive branch).

Winning strategies

Here a crucial role has the fact: who starts an initial game. Indeed, if Proponent starts an initial game, then she can continuously open a new copy of the initial game and hence to keep the rights on move infinitely long. Of course, in that case Opponent can not have a winning strategy in the compound game (but that also does not give a winning strategy for Proponent). If Opponent would have a winning strategy at the initial game (together with first move) then obviously he will have it in the compound game. In short instruction for that operator Blass (1992) wrote that for Opponent enough to win one of the opened game, whereas Proponent has to win all of them.

Proponent has a winning strategy in compound game if she has the first move and has a winning strategy in the initial game.

Definition 5.2 ***Repetition "why not"***

$\overline{R}\,(A) = (M_{\overline{R}\,(A)}, \lambda_{\overline{R}\,(A)}, P_{\overline{R}\,(A)}, W_{\overline{R}\,(A)})$

Where:

- $M_{\overline{R}\,(A)} = M_{R(A)}$
- $\lambda_{\overline{R}\,(A)}$ labelling function: it is Proponent's turn to move if and only if it is her turn to move in all opened games, otherwise it is Opponent's turn to move (note that if Opponent would have a first move in the initial game, that Proponent will not have a winning strategy, because Opponent can continuously open a new game)
- $P_{\overline{R}\,(A)}$ is the set of valid positions (admissible moves) that constructed as follows:
  1) If it is Proponent's turn to move then she can move in any possible direction (during her turn she is allowed to move in all of the components, and hence she can always switch the game)
  2) If it is Opponent's turn to move then he can move only in the components where it is his turn to move (note that if it is his turn to move in several components then he can switch the games and open a new game)
- $W_{\overline{R}\,(A)}$ is the set of all winning for Proponent branches that equal to union of the winning branches of the initial game (i.e. to win the game for Proponent enough to win in one of the components of the resulting multiplicative branch).

Winning strategies

Proponent and Opponent switch the roles in comparison with the repetition "of course" games.

5.1.2 Trial-mistake strategies

Copy strategy discussed in the previous chapter is based on the possibility to switch the games, which is hidden in game structure in construction of the set of admissible sequences of moves. The possibility of communication between the games is also important.

Trial-mistake strategy – another example of structural φ^I -strategy – is based on the idea of searching through the game tree. Trial-mistake strategy also concerns undetermined games. If we have parallel game $\alpha\otimes\alpha\otimes\ldots\otimes\alpha$ (or $\alpha\oplus\alpha\oplus\ldots\oplus\alpha$) with amount of opened games α close to the number of final states of the game α, then we can describe trial-mistake strategy of Opponent (Proponent) as instruction for an unique playing of game α in each opened game α. Hence trial-mistake strategy allows combing the game tree and finding the winning branches.

As it is easy to see the combination $\alpha\otimes\alpha\otimes\ldots\otimes\alpha$ ($\alpha\oplus\alpha\oplus\ldots\oplus\alpha$) could be changed by the parallel game with repetition operator "of course" ("why not"). It would be interesting to adapt these operators within the game model.

5.2 Parallelism of Linear Logic and of Process Algebra

In previous chapters we analysed operators governing of parallel processes in Linear Logic and Process Algebra, and introduced the operators for Game Logic. In this section have a short discussion focussing on switching that is a form of communication.

5.2.1 Non-communicative processes

In Linear Logic we can call instantiation of parallel operator non-communicative if it does not use any type of interactions between the games like copy strategy. As we saw, Process Algebra outlines such types of processes precisely.

First of all it is seems that the Process Algebra can better express one-player games as opposed to two-player games. We will keep different point of view, and assume that Process Algebra can describe two-player games, which are played by one player. However, since one-player games are the simplest, we begin the description of non-communicative instantiation of tensor and par from such type of game.

It is clear that tensor (par) in a game where only Opponent (Proponent) has moves can be expressed by free merge. Suppose that in games α and β depicted in Figure 4.2 all the moves are performed by Opponent only, then tensor game $\alpha\otimes\beta$ can be written through free merge if we will change the description level from names of games α and β to the names of moves m_0, m_1, m_2, n_0, n_1. In this instance tensor game can be understood as $(m_0 + m_1)\, m_2 \parallel (n_0 + n_1)$.

In two-player tensor (par) game Proponent (Opponent) can not switch between the games. It means that for rewriting of her/his moves we can not use free merge because it would allow her/him to switch. Moreover, if we would rewrite the moves of Opponent (Proponent) by using free merge, then by axioms of Process Algebra we have to execute her/his moves in both subgames, but it is non-convenient. Besides, we would have problems to point out that Proponent (Opponent) has to make her/his next move in active game. However, alternative and sequential compositions can describe non-communicative instantiation of two-player game. It is easy to see if look at tensor (par) game tree.

5.2.2 Communicative processes

In Linear Logic we can call instantiation of parallel operator communicative if it use any type of interactions between the games like copy strategy. In Process Algebra communicative merge allows transmitting data between the processes.

In copy strategy Proponent and Opponent plays symmetrical roles as in tensor so in par. So, to express tensor or par moves through communication merge we have to decompose the games on moves and then to write mirrored moves from different sides of communication merge. The draw of game will look like a chain of communication merges connected by sequential composition.

5.2.3 Extension of communication merge

In case of compound games we would have to connect more than two games. For this purpose we have to extend communication function and communication merge on multiple process problems. Previousely they described only interaction of send/read type similar to copy strategy in Linear Logic.

We redefine communication function, which connects several processes in one. It is the partial mapping γ: $A\times\ldots\times A\rightarrow A$, where A is the set of the atomic actions. As an example of such kind of interaction we can give screen picture. Consider light flows on a computer screen. To make a composed colour on the screen we have to combine three different colours at one moment.

5.3 Laws of Multiplicative Linear Logic and proposed semantics

Abramsky and Jagadeesan (1994) introduced sound and complete calculus for their game semantics. In this section we explore if these laws are sound to our semantics of finite determined games. The laws can be rewritten by axioms (5.1) – (5.6). In Linear Logic semantics there is no φ^I-strategy. The laws apply to the winning strategies of Proponent. Therefore, during the proof of soundness we can refer to the axioms introduced in chapter 4, all of which constituents have a form $\langle\pi\rangle\varphi$, and keep in mind that we have a deal with winning strategies but not φ^I-strategy. It is a reason why we will talk about analogues of the axioms during the proof of soundness.

$\vDash \alpha\oplus\alpha^d$	an analogue of Identity law	(5.1)
$\vDash (\gamma\oplus\delta) \rightarrow (\delta\oplus\gamma)$	an analogue of Exchange law	(5.2)
$\vDash (\gamma\oplus(\delta\oplus\varepsilon)) \rightarrow ((\gamma\oplus\delta)\oplus\varepsilon)$	an analogue of Exchange law	(5.3)
$\vDash ((\gamma\oplus\alpha) \wedge (\delta\oplus\beta)) \rightarrow (\gamma\oplus(\delta\oplus(\alpha\otimes\beta)))$	an analogue of Tensor law	(5.4)
$\vDash ((\gamma\oplus\alpha) \wedge (\delta\oplus\alpha^d)) \rightarrow (\gamma\oplus\delta)$	an analogue of Cut law	(5.5)
$\vDash (\gamma \wedge \delta) \rightarrow (\gamma\oplus\delta)$	an analogue of Mix law	(5.6)

Theorem 5.1

Axioms (5.1) – (5.6) are sound modulo φ^I-strategy with respect to arbitrary φ^I-zero-sum interpretation of game structures given in the previous chapter.

Proof

(5.1) $\vDash \alpha\oplus\alpha^d$

Proponent has a winning strategy in such a par game. For Proponent it is enough to win only one of the subgames. Defined game structure allows Proponent to play both subgames; then if she/he will choose the same track of play in both components, it will guarantee a winning exactly of one of them.

(5.2) $\models (\gamma \oplus \delta) \to (\delta \oplus \gamma)$

Analogue arguments as in proof of axioms (4.8)

(5.3) $\models (\gamma \oplus (\delta \oplus \varepsilon)) \to ((\gamma \oplus \delta) \oplus \varepsilon)$

Analogue arguments as in proof of axioms (4.12)

(5.4) $\models ((\gamma \oplus \alpha) \wedge (\delta \oplus \beta)) \to (\gamma \oplus (\delta \oplus (\alpha \otimes \beta)))$

(i) $((\gamma \oplus \alpha) \wedge (\delta \oplus \beta)) \leftrightarrow ((\gamma \oplus \alpha) \otimes (\delta \oplus \beta))$ by an analogue of (4.21)

(ii) $(\gamma \oplus (\delta \oplus (\alpha \otimes \beta))) \leftrightarrow ((\gamma \oplus \delta \oplus \alpha) \otimes (\gamma \oplus \delta \oplus \beta))$ by an analogues of (4.21), (4.22), and (4.26)

(iii) $(\gamma \oplus \alpha) \to (\gamma \oplus \delta \oplus \alpha)$ if Proponent has a winning strategy in $(\gamma \oplus \alpha)$ then she/he can avoid to play δ, and hence to win $(\gamma \oplus \delta \oplus \alpha)$

(iv) $((\gamma \oplus \alpha) \wedge (\delta \oplus \beta)) \to (\gamma \oplus (\delta \oplus (\alpha \otimes \beta)))$ from (i), (ii), (iii)

(5.5) $\models ((\gamma \oplus \alpha) \wedge (\delta \oplus \alpha^d)) \to (\gamma \oplus \delta)$

$((\gamma \oplus \alpha) \wedge (\delta \oplus \alpha^d))$ means that Proponent has to win both $(\gamma \oplus \alpha)$ and $(\delta \oplus \alpha^d)$. Further we have to consider three cases:

(i) Suppose Proponent has a winning strategy in α. Then she/he has no a winning strategy in α^d. Then to win $(\delta \oplus \alpha^d)$ she/he has to have a winning strategy in δ, or δ has to be equal α to apply a copy strategy. But we assumed that Proponent has a winning strategy in α, then in any case under the assumption Proponent will have a winning strategy in δ and hence in $(\gamma \oplus \delta)$.

(ii) The case with a winning strategy for Proponent in α^d is similar.

(iii) Assume Proponent has no winning strategies either in α or in α^d, then to win $(\gamma \oplus \alpha)$ and $(\delta \oplus \alpha^d)$ she/he has to have winning strategies for γ and δ, and hence for $\gamma \oplus \delta$, or to have copy strategies in both $(\gamma \oplus \alpha)$ and $(\delta \oplus \alpha^d)$. This last means that γ is equal to α^d, and δ is equal to α. Hence in $(\gamma \oplus \delta)$ Proponent has a winning strategy. It is a copy strategy.

(5.6) $\models (\gamma \wedge \delta) \to (\gamma \oplus \delta)$

Follows from an analogue of (4.19)

$\times$

5.4 Conclusion

In the thesis we introduced an Extension of Game Logic with parallel operators, which we constructed by analogues with multiplicative operators tensor and par of Linear Logic.

As we found, it is not so easy, as it seems at first sight, to extend Game Logic with the parallel operators of Linear Logic, and especially of Process Algebra, even without a proof of completeness. The main reason is that if Linear Logic has a kind of classical or extensional semantics, then Game Logic has an intensional one. Distinction between these two semantics is comparable to the distinction between winning and φ^I-strategies. Thus, when we are in the game model of Linear Logic, it is impossible that both players would have winning strategies at some of the states, because it would mean that a formula is true and false simultaneously. Whereas, for the model of Game Logic, it is possible that both players would have φ^I-strategies at some state, because it would just express that neither player can achieve a state where φ is not valid. This gives different validities of axioms for Game Logic as opposed to Linear Logic.

Moreover, we found that finite perfect information games do not add anything new to the semantics. Tensor and par in that case are equal to the known operators of choice! The power of parallelism can be achieved only for infinite or imperfect information games. On the other hand, this resembles the situation in Process Algebra, where in the finite case, non-communicative parallel composition can be reduced to alternative and sequential ones.

Iouri Netchitailov would like to thank:
Marc Pauly and Johan van Benthem for brilliant supervising, very helpful comments, and interesting discussions;
Ingrid van Loon and Dick de Jongh for organisation assistance;
Staff of Institute of Logic Language and Computation for educational and interesting classes;
University of Amsterdam for financial support;
All friends and relatives, who inspire my existence;

We gratefully acknowledge Eveline Visser for useful discussion and linguistic comments.

References

Abramsky, S., Jagadeesan, R., 1994, The Journal of Symbolic Logic, 59 (2), 543

Apt, K. R., 2000, *Principles of Constraint Programming*, http://www.cwi.nl/~apt/krzysztof.html

Baeten, J.C.M., Verhoef, C., 199X, in *"Handbook of Logic in Computer Science"*, 4, 149

van Benthem, J., 1999B, Games and Strategies inside Elementary Logic, in *"Proceedings 7th Asian Logic Conference*", Taiwan

van Benthem, J., 1999A, *"When are Two Games the Same?"*, http://turing.wins.uva.nl/~johan/

Bergstra, J.A., Klop, J.W., 1984, *Information and Control*, 60 (1/3), 109

Binmore, K., 1992, *"Fun and Games. A Text on Game Theory"*, ISBN 0-669-24603-4

Blass, A., 1992, Annals of Pure and Applied Logic, 56, 183

Fokkink, W., 2000, *"Introduction to Process Algebra"*, Springer-Verlag (Berlin)

Gale, D., Stewart, F.M., 1953, in *"Contributions to the Theory of Games"*, Ann. Of Math. Stud., 28, 245 (Princenton Univ. Press, Princenton, NJ)

Girard, J.Y., 1987, Theoret. Compu. Sci., 50 (1), 1

Lorenzen, P., 1959, in *"Infinitistic Methods"*, PWN, Warsaw, 193

Milner, R., 1980, *"A Calculus of Communicating Systems"*, LNCS 92, Springer-Verlag (Berlin)

Moller, F., 1990, in *"Proceedings 17th Colloquium on Automata, Languages and Programming"* (ICALP'90), Warwick, LNCS 443, 752, Springer-Verlag (Berlin)

Neumann, V., Morgenstern, 1944, *The Theory of Games and Economic Behavior*

Parikh, R., 1985, in *"Annals of Discrete Mathematics"*, 24, Elsevier

Pauly, M., 1999, in *"Position papers tutorials LGS'99"*, editor Swart, H., Tilburg University, The Netherlands, 72