Available online at www.sciencedirect.com

SCIENCE DIRECT

ELSEVIER Electronic Notes in Theoretical Computer Science 88 (2004) 21–37

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A–maze–ing Esterel

Joaquin Aguado and Michael Mendler[1],[2]

Fakultät für Wirtschaftsinformatik und Angewandte Informatik, Universität Bamberg, Germany

Gerald Lüttgen[3]

Department of Computer Science, The University of York, U.K.

Abstract

This paper shows that the kernel fragment of Esterel corresponding to combinational circuits admits a natural game–theoretic interpretation. Technically, combinational Esterel programs are mapped into finite two–player games in such a way that the standard *must*– and *cannot*–analysis of signal statuses is reflected in the computation of winning strategies. The novel game–theoretic approach complements the existing behavioral, operational, circuit–based, and model–theoretic accounts of Esterel's semantics and offers a new didactic perspective for familiarizing students and engineers with this intricate constructive semantics.

Keywords: Synchronous languages, Esterel, constructive semantics, causality cycle, game theory

1 Introduction

The classical theory of games, originally developed in descriptive set theory, has recently emerged as a surprisingly versatile mathematical tool in the semantics of programming languages [1]. The power of the games model rests on its ability to handle combinatorially complex situations, such as the alternate nesting of quantifiers, in a natural and intuitive fashion [8]. Perhaps the most prominent recent example of successful application is the game–theoretic

[1] Email: joaquin.aguado@wiai.uni-bamberg.de.

[2] Email: michael.mendler@wiai.uni-bamberg.de.

[3] Email: luettgen@cs.york.ac.uk. Research supported by EPSRC grant GR/M99637.

doi:10.1016/j.entcs.2003.05.006

solution of the full–abstraction problem for the functional language PCF [5], which had been open for a long time. This has led to new approaches in the field of control–flow analysis [7], integrating imperative, object–oriented, higher–order functional, and concurrent features.

Games are a convincing metaphor not only in functional programming but also in the field of reactive–systems modeling. This is because the interaction between a reactive system and its environment has a strong analogy in the moves between a player and his or her opponent in a simple two–player maze game. This interaction problem is then solved by providing a winning strategy that in turn may be understood as a system reaction. In this paper we report on a novel application of this metaphor to the specific interaction problem that arises in synchronous programming under the synchrony hypothesis, namely the characterization of *present* and *absent* signals within a system's reaction under a given environment. Using Berry's language *Esterel* [3] as an example, we prove that the underlying intricate constructive semantics of reactions can be captured in a very natural game–theoretic manner.

Consider a two–player maze game, where the board consists of one–way *corridors* that connect *rooms* and the game figure is just a single token. Corridors can be of two types: *visible* and *secret*. Placing the token in some arbitrary room, the starting player, say Jaakko, may move the token from one room to the next through arbitrarily many secret corridors; however, as soon as Jaakko moves through a visible corridor, his turn ends. Control passes on to the opponent, say Leon, who may continue in a similar fashion from the current position of the token on the board.[4] If the token gets stuck in a *dungeon*, i.e., a room with no outgoing corridors, the current player loses and the opponent wins. Hence, the objective of the game is to drive the opponent into a *dungeon*. Obviously, a game board can be modeled as a finite directed graph, such as the one depicted in Fig. 1, where nodes are rooms, solid edges represent visible corridors, and dashed edges represent secret corridors.

Given a game board and choosing an initial room for the token, this room may now be classified according to whether the starting player Jaakko (i) has the possibility always to win (if he plays cleverly), (ii) must always lose (no matter how cleverly he plays), or (iii) can at best reach a draw by ensuring that Leon can never force him into a dungeon. For instance, using the board $M_{\texttt{ex}}$ illustrated in Fig. 1 and initially placing the token in room t_5, Jaakko can move the token to room t_6 through a visible corridor, thereby handing over control to Leon. Now Leon can *only* move the token into dungeon t_0, again through a

[4] The players are named after the logicians Jaakko Hintikka, who pioneered the field of semantic games for logic, and Leon Henkin, who first introduced game–theoretic interpretations of quantifiers.

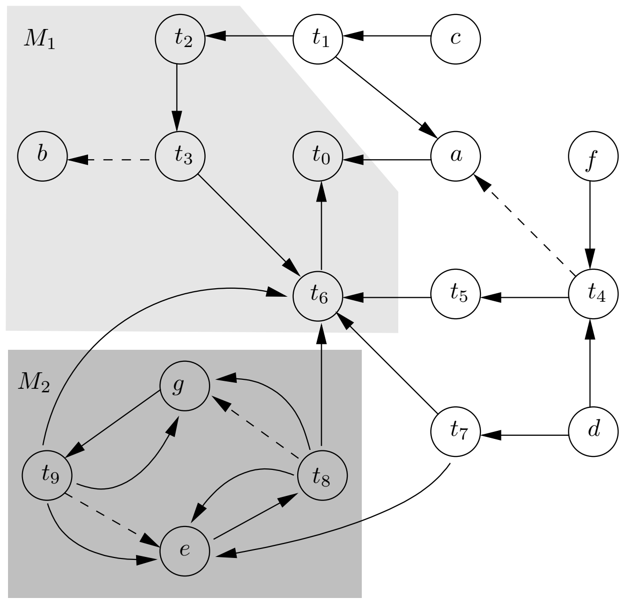

Fig. 1. Example game board M_{ex}.

visible corridor, such that control passes back to Jaakko who instantaneously loses. This implies that room t_5 is a *losing position* (for the starting player Jaakko). However, if instead the token is initially placed in room t_4, then Jaakko has two strategies for winning. On the one hand, Jaakko may move the token through a visible corridor to room t_5 where, as we have just seen, player Leon will necessarily lose after two more turns. On the other hand, Jaakko may use the secret corridor to move the token into room a and in the same turn further into dungeon t_0 via a visible corridor, thus winning the game. Jaakko is said to have a *winning strategy* from room t_4, and t_4 is referred to as a *winning position.* However, a game may also end in a draw. For example, suppose the token is placed initially in room t_8. Then Jaakko has several alternatives, but one of these leaves him in the hands of Leon. Indeed, if Jaakko moves the token to room t_6, Leon can place the token into dungeon t_0. Observe that if the token is instead placed in t_9, the situation is similar in the sense that moving the token in room t_6 will result in losing the game. Now, assuming that both players want to win and that they both know that placing the token in room t_6 is the worst option, it follows that they will keep moving the token all the time through the dark–shaded part of Fig. 1, i.e., through sub–maze M_2. Since inside M_2 it is always possible to avoid room t_6, the game can continue indefinitely in this fashion leading to a draw, whence rooms e, g, t_8 and t_9 are referred to as a *draw positions.*

Maze Games and Esterel.

The objective of this paper is to show that programs written in the kernel fragment of Esterel corresponding to combinational circuits can be understood very naturally as maze game boards, where signals are represented by rooms. Intuitively, the presence and absence of signals in the reaction instant that is described by a combinational program P, is "negotiated" between the system (the starting player) and its environment (the opponent). The system tries to prove a signal's presence and the environment its absence. Thus, signal s must (cannot) be emitted in P if and only if room s in the maze M associated with P is a winning (losing) position. If the status of signal s is undefined, then and only then is room s a draw position.

Technically, strategies within M correspond to the *must*– and *cannot*–analysis for P, which is at the heart of Esterel's behavioral semantics [3]. A simple way to make the connection is to read each *visible* (*secret*) corridor as a *present–else statement* (*present–then statement*). We then get an exact correlation between Esterel's declarative computation of *must*– and *cannot–sets* of signals [3] and an inductive computation of winning and losing positions in the game graph. We illustrate this correspondence using the sub–maze M_1 in our example of Fig. 1. The program P_1 associated with M_1 is:

$$(\texttt{present } t_3 \texttt{ else emit } t_2 \texttt{ end}) \parallel (\texttt{present } b \texttt{ then emit } t_3 \texttt{ end}) \parallel$$
$$(\texttt{present } t_6 \texttt{ else emit } t_3 \texttt{ end}) \parallel (\texttt{present } t_0 \texttt{ else emit } t_6 \texttt{ end}) .$$

We reason along the fixed–point computation of P_1's declarative semantics and start with the empty environment in which no signal is known to be present or absent, whence $must^0 = cannot^0 = \emptyset$. Since no `emit` is unguarded, the first iteration yields no present signals, i.e., $must^1 = \emptyset$; but since there are no `emit` statements for b or t_0, we get $cannot^1 = \{b, t_0\}$ immediately. In game terms this corresponds to identifying both rooms b and t_0 in M_1 as positions in which the starting player loses right away. The fact that t_6 is connected to t_0 by a *visible* corridor means that the starting player now has a strategy to win t_6, because he or she can move into t_0 where his or her opponent loses. In the computation of Esterel's declarative semantics for P_1, this is the second iteration step: since t_0 is known as absent, the emit statement in `present` t_0 `else emit` t_6 `end` is executed and t_6 becomes present. We thus get $must^2 = \{t_6\}$ and $cannot^2 = \{b, t_0\}$. Additionally, we know that the statement `present` b `then emit` t_3 `end` is not executed in the current instant. In the game, this amounts to marking the secret corridor from t_3 to b as useless for any winning strategy for t_3. There is no point for any player in going across to b since the player keeps his or her turn and thus loses in b. It may still be possible to win by moving from t_3 to t_6. However, with the extra information just obtained, namely that t_6 is a winning position, we conclude that t_3 is in fact a losing position. For moving from t_3 to t_6 does not help either since the opponent would get the turn in t_6 and win. In the Esterel approximation sequence, t_3 indeed enters the *cannot* set in the third iteration step: $cannot^3 = \{b, t_0, t_3\}$. This is clear since $t_6 \in must^2$ and $b \in cannot^2$, whence the only two statements that could emit t_3 in P_1 are both switched off. The *must* set does not change, whence $must^3 = must^2 = \{t_6\}$. The fourth iteration step identifies t_2 as emitted from the fact that $t_3 \in cannot^3$. For this means, the statement `present` t_3 `else emit` t_2 `end` is executed. In game terms, room t_2 is clearly a winning position as t_3 is a losing position. Hence, we obtain $must^4 = \{t_6, t_2\}$ and $cannot^4 = cannot^3 = \{b, t_0, t_3\}$ as the fixed point of Esterel's constructive analysis for P_1. To sum up, we see that $must^{n+1}$ ($cannot^{n+1}$) is the set of rooms that can be won (must be lost) by the starting player in at most n moves.

The example in Fig. 1 also illustrates how constructiveness of Esterel reactions is reflected in the game model. We have seen above that the shaded area M_2 of $M_{\texttt{ex}}$ contains only draw positions. The associated Esterel program P_2 can be written as a parallel composition of eight present statements, as suggested above, or equivalently in a more compact form as

$$
\begin{aligned}
P_2 := {} & \texttt{present}\ g\ \texttt{then present}\ e\ \texttt{then}\\
& \quad (\texttt{present}\ g\ \texttt{else emit}\ e\ \texttt{end}) \parallel\\
& \quad (\texttt{present}\ e\ \texttt{else emit}\ g\ \texttt{end})\\
& \texttt{end end}\,,
\end{aligned}
$$

where the two rooms t_8 and t_9 are no longer represented as signals but are implicit in the nested present statements. As an aside, we will see below that our intermediate states t_i in game graphs are necessary to express conjunctive behavior. The *must*– and *cannot*–analysis for P_2 indeed leaves signals e and g constructively undecided. To justify emission of either e or g, both e and g would have to be present in the first place to activate the outmost present statements, which is causally unreasonable. At the same time, they must be absent to switch on the inner emit statements, which is contradictory overall. We cannot justify the absence of e and g, causally, either. For example, in order to deactivate the inner emits through one of the outer guarding present conditions we would need that one of e and g is absent, which is also causally problematic. There is a second possibility for the absence of e and g, namely that the inner emit statements are both switched off. This however requires both signals to be present, which is again a contradiction. Overall, there is only one logically coherent solution, namely that both e and g are absent, yet this solution is not causal. Since this is the only logically coherent solution, the *logical behavioral semantics* of Esterel [3] would return this as the response, whereas the constructive semantics rejects it. In our maze game, a logically coherent solution amounts to a "speculative" assignment of 0 (losing) and 1 (winning) markings to rooms so that (i) a room is marked 1 exactly if one of its successor rooms that is accessible via a visible corridor is marked 0, or if a successor room connected via a secret corridor is marked 1; and (ii) a room is marked 0 if all rooms connected via visible corridors are marked 1 and if all rooms connected through a secret corridor are marked 0. In Fig. 1, $e = g = 0$ and $t_8 = t_9 = 1$ is the only logically coherent marking for M_2. Although this might suggest that both e and g are losing positions for the starting player, it is clear that this cannot be realized by any finite strategy of the opponent.

2 Mazes and Maze–Game Semantics

This section formalizes our two–player maze games and also provides an alternative denotational characterization of the operational notion of a *winning position*. The reader may find some background material on classical games in [10,12]. The only change in our setting over the classical definitions is that we (i) allow for two types of transitions in game graphs, i.e., visible and secret transitions, and (ii) admit draw positions. The latter feature is in contrast to the classical games used in automata theory and descriptive–set theory [12], where the absence of a winning strategy for one player automatically implies the existence of a winning strategy for the other.

Formalizing Mazes.

Mazes are essentially finite graphs with two kinds of directed edges, namely *visible* and *secret* edges. For our purposes, it is convenient to formally represent these graphs as *systems of unfolding rules* $M := (x \Leftarrow m_x)_{x \in \mathcal{V}}$ in a language of mazes, for some finite set $\mathcal{V}$ of variables representing rooms and maze terms m_x. *Maze terms* are defined in a process–algebraic fashion, which provides us with sufficient structure for proving the paper's main results. The syntax of maze terms is given by the following BNF:

$$m ::= x \mid 0 \mid \iota.m \mid \tau.m \mid m + m \,.$$

Intuitively, 0 represents a dungeon, $\iota.m$ ($\tau.m$) represents a room with a visible (secret) corridor to room m, and $m_1 + m_2$ represents a room that merges rooms m_1 and m_2, respectively. In the remainder, we let $\mathcal{M}$ stand for the set of all maze terms.

For each room x in any given maze M we would like to determine whether it is a winning position (for the starting player). The game–theoretic semantics of maze M requires the introduction of a *labeled transition system* $\langle \mathcal{M}, \{\iota, \tau\}, \longrightarrow \rangle$, where $\mathcal{M}$ is the set of states (or *rooms*), $\{\iota, \tau\}$ is the alphabet with ι encoding a *visible action* and τ a *secret action*, and $\longrightarrow \subseteq$

$\mathcal{M} \times \{\iota, \tau\} \times \mathcal{M}$ is the transition relation representing valid moves (or *corridors*) between rooms. The transition relation is defined by the following rules, where γ ranges over $\{\iota, \tau\}$:

$$\frac{-}{\gamma.m \xrightarrow{\gamma} m} \qquad \frac{m_1 \xrightarrow{\gamma} m_1'}{m_1 + m_2 \xrightarrow{\gamma} m_1'} \qquad \frac{m_2 \xrightarrow{\gamma} m_2'}{m_1 + m_2 \xrightarrow{\gamma} m_2'} \qquad \frac{m \xrightarrow{\gamma} m'}{x \xrightarrow{\gamma} m'}\ x \Leftarrow m\ .$$

Essentially, this labeled transition system reflects the game graphs of Sec. 1, with dungeons being traps that have no outgoing transition. In the following, we write $m \longrightarrow$ for $\exists m'\, \exists \gamma.\ m \xrightarrow{\gamma} m'$. Note that operators ".", "+", and "$\Leftarrow$" correspond to the process–algebraic operators *prefix*, *choice*, and *recursion*.

Example 2.1 Let us specify the maze in Fig. 1 relative to the program's signals, also called *named rooms*, i.e., $\mathcal{V} = \{a, b, c, d, e, f, g\}$. The other rooms $\{t_0, t_1, \ldots, t_9\}$ are referred to as *unnamed rooms* and are represented implicitly as sub–terms in the corresponding system of unfolding rules $M_{\mathsf{ex}} := (x \Leftarrow m_x)_{x \in \mathcal{V}}$ with $a \Leftarrow \iota.0$, $b \Leftarrow 0$, $c \Leftarrow \iota.(\iota.a + \iota.\iota.(\tau.b + \iota.\iota.0))$, $d \Leftarrow \iota.(\tau.a + \iota.\iota.\iota.0) + \iota.(\iota.\iota.0 + \iota.e)$, $e \Leftarrow \iota.(\iota.e + \iota.g + \tau.g + \iota.\iota.0)$, $f \Leftarrow \iota.(\tau.a + \iota.\iota.\iota.0)$, $g \Leftarrow \iota.(\iota.g + \iota.e + \tau.e + \iota.\iota.0)$. Observe that, for any $x \Leftarrow m_x$, applying the operational rules to m_x results in the part of the graph starting from room x. Specifically, for $f \Leftarrow m_f$ and $m_f = \iota.(\tau.a + \iota.\iota.\iota.0)$, the first ι in the term corresponds to the visible corridor connecting f and the unnamed room $t_4 = \tau.a + \iota.\iota.\iota.0$. From t_4, either a secret corridor τ can be taken to room a, or a path of three visible corridors can be followed reaching dungeon t_0.

Playing the Maze Game.

We now turn our attention to the game–theoretic semantics of our two–player maze game. For convenience, we will name the players simply A and B. We begin by defining the notions *dungeon*, *path*, and *turn*. Firstly, room m is a dungeon if $m \not\longrightarrow$. Secondly, a *path* π through a maze M is a sequence of transitions $(m_i \xrightarrow{\gamma_i} m_{i+1})_{0 \le i < k}$, where $k \in \mathbb{N} \cup \{\omega\}$ is referred to as the *length* $|\pi|$ of π. We say that π is finite in case $k < \omega$; otherwise, π is infinite. A path π is *maximal* if it is either infinite, or if it is finite and $m_{|\pi|} \not\longrightarrow$. We abbreviate π's finite prefix $(m_i \xrightarrow{\gamma_i} m_{i+1})_{0 \le i < j}$ of length $j \in \mathbb{N}$ by π^j. Finally, given a finite (prefix of a) path π and assuming player A always starts off the game, we can determine the player *turn*(π) whose turn it is in the final room $m_{|\pi|}$ as follows, where $\overline{A} := B$ and $\overline{B} := A$:

$$turn(\pi) := \begin{cases} turn(\pi') & \text{if } |\pi| > 0 \text{ and } \pi = \pi' \cdot \xrightarrow{\tau} \\ \overline{turn(\pi')} & \text{if } |\pi| > 0 \text{ and } \pi = \pi' \cdot \xrightarrow{\iota} \\ A & \text{otherwise, i.e., } |\pi| = 0 \ . \end{cases}$$

A maze play is determined by the players' strategies. A *strategy* is a (partial) function $\alpha : \mathcal{M} \rightharpoonup \{\iota, \tau\} \times \mathcal{M}$ such that, for all $m \in \mathcal{M}$, if $\alpha(m) = (\alpha_1(m), \alpha_2(m))$ is defined then $m \xrightarrow{\alpha_1(m)} \alpha_2(m)$. Note that a strategy of a player does not depend on the opponent's strategy or on a play's history. Given strategies α and β for players A and B, respectively, the play $play_M(\alpha, \beta, m)$ in maze M starting in room m with player A is a maximal path $\pi = (m_i \xrightarrow{\gamma_i} m_{i+1})_{0 \leq i < k}$ in M such that $m_0 = m$, $m_{i+1} = \alpha_2(m_i)$ and $\gamma_i = \alpha_1(m_i)$ if $turn(\pi^i) = A$, or $m_{i+1} = \beta_2(m_i)$, $\gamma_i = \beta_1(m_i)$ if $turn(\pi^i) = B$.

The operational semantics of maze $M = (x \Leftarrow m_x)_{x \in \mathcal{V}}$ considers, for each room x, whether player A or B has a winning strategy, or neither of them. Intuitively, A (B) has a winning strategy, if he or she is always able to drive player B (A) into a dungeon, no matter which strategy B (A) employs and always assuming that player A starts off the game. Formally, player A has a winning strategy for room x in M if $\exists \alpha \, \forall \beta \, . \; |play_M(\alpha, \beta, x)| < \omega$ and $B = turn(play_M(\alpha, \beta, x))$. Dually, player B has a winning strategy for room x in M if $\exists \beta \, \forall \alpha \, . \; |play_M(\alpha, \beta, x)| < \omega$ and $A = turn(play_M(\alpha, \beta, x))$. If the selected starting player A has a winning strategy for room x, then x is simply called a *winning position*. If player B has a winning strategy, then x is a *losing position*. If both players can always avoid dungeons, thus engaging in infinite plays, neither player wins and the play ends in a draw. Accordingly, a position that is neither a winning nor a losing position is referred to as a *draw position*.

Example 2.2 Consider again maze M_{ex} of Fig. 1. Suppose that the strategy α of player A is chosen such that A moves the token, whenever possible, to a named room from $\mathcal{V}$ through a secret corridor. Otherwise, he chooses a room t_i with the smallest i. The strategy β of player B is that B prefers rooms of the form t_i with the smallest i over any $\mathcal{V}$. Yet, overall, B prefers secret corridors whenever there is one. For example, $\alpha(d) = t_4$, $\alpha(t_6) = t_0$, $\beta(t_4) = a$, $\beta(a) = t_0$, $\beta(t_7) = t_6$. Thus, the play $play_{M_{\mathsf{ex}}}(\alpha, \beta, d)$ is the path $d \xrightarrow{\iota} t_4 \xrightarrow{\tau} a \xrightarrow{\iota} t_0$, along which A loses. In contrast, if player A chooses t_7 in the first turn, B would have selected the move to t_6, from where A moves to the dungeon t_0 and player B loses. The play in that case would be $d \xrightarrow{\iota} t_7 \xrightarrow{\iota} t_6 \xrightarrow{\iota} t_0$. The best player B can do under this new strategy of A is to set $\beta(t_7) = e$, while otherwise keeping his preference for secret corridors. In that case the play amounts to a draw along the infinite path $d \xrightarrow{\iota} t_7 \xrightarrow{\iota} e \xrightarrow{\iota} t_8 \xrightarrow{\tau} g \xrightarrow{\iota} t_9 \xrightarrow{\tau} e \xrightarrow{\iota} t_8 \cdots$.

Denotational Characterization.

The semantics of mazes in terms of winning and losing positions may also be captured denotationally. The denotational approach relies on two predicates *win* and *lose* that take as parameters a maze term m and an *environment* X. Adapting notational conventions from Esterel, we define an environment X as a set of *signed* variables x^+ and x^-, with the meaning that plus–tagged (minus–tagged) variables represent rooms that are known to be winning (losing) positions. Since a single room cannot be both a winning and a losing position, an environment must not contain both x^+ and x^-, for any x. Predicate *win* (*lose*) now holds for m and X if m corresponds to a winning (losing) position relative to X, respectively. Formally, these predicates are defined as the least predicates respecting the following rules:

$$\begin{array}{lllll}
\mathit{lose}(0,X) & \mathit{win}(x,X) \text{ if } x^+ \in X & & \mathit{lose}(x,X) \text{ if } x^- \in X & \\
 & \mathit{win}(\iota.m,X) \text{ if } \mathit{lose}(m,X) & & \mathit{lose}(\iota.m,X) \text{ if } \mathit{win}(m,X) & \\
 & \mathit{win}(\tau.m,X) \text{ if } \mathit{win}(m,X) & & \mathit{lose}(\tau.m,X) \text{ if } \mathit{lose}(m,X) & \\
 & \multicolumn{4}{l}{\mathit{win}(m_1+m_2,X) \text{ if } \mathit{win}(m_1,X) \ \mathit{or} \ \mathit{win}(m_2,X)} \\
 & \multicolumn{4}{l}{\mathit{lose}(m_1+m_2,X) \text{ if } \mathit{lose}(m_1,X) \ \mathit{and} \ \mathit{lose}(m_2,X)\,.}
\end{array}$$

Intuitively, the dungeon 0 is always a losing position. Room $m_1 + m_2$ is a winning position if at least one of m_1 or m_2 is, since $m_1 + m_2$ essentially gives a free choice to the leading player whether to continue with m_1 or m_2. Dually, $m_1 + m_2$ is a losing position if both m_1 and m_2 are. Room $\iota.m$ can only be left by the visible corridor ι to m, thereby giving control to the opponent. Hence, $\iota.m$ is a winning (losing) position for the leading player if m is a losing (winning) position for the opponent. Traversing a secret corridor does not change a player's turn, whence $\tau.m$ is a winning (losing) position for the leading player if m is. Thus, by an appropriate choice of visible and secret corridors out of a room x we can make the winning (losing) predicate for x an arbitrary disjunctive (conjunctive) combination of negated and non–negated winning conditions of the immediate successor rooms. This is useful for normal–form representations and explains why we introduce secret corridors into the games model. We now define a function on environments:

$$\mathit{maze}(M)(X) := \mathit{win}(M,X)^+ \cup \mathit{lose}(M,X)^- \,.$$

Here, $\mathit{win}(M,X)$ denotes $\{x \in \mathcal{V} \mid \mathit{win}(m_x,X),\ x \Leftarrow m_x \text{ in } M\}$ and $\mathit{lose}(M,X)$ for $\{x \in \mathcal{V} \mid \mathit{lose}(m_x,X),\ x \Leftarrow m_x \text{ in } M\}$. Additionally, for any subset $V \subseteq \mathcal{V}$ of variables, V^+ denotes the set $\{x^+ \mid x \in V\}$ and V^- the set $\{x^- \mid x \in V\}$. It can easily be proved that function $\mathit{maze}(M)$ is monotonic. Hence, the least fixed point $\mu\mathit{maze}(M)$ of $\mathit{maze}(M)$ exists, which is taken to be the denotational semantics of maze M. Moreover, because our universe of variables is finite, one may iteratively compute $\mu maze(M) = \bigcup_{i \in \mathbb{N}} maze(M)^i(\emptyset)$.

Example 2.3 Let us obtain the sets of winning, losing, and draw positions for maze $M_{\texttt{ex}}$ of Fig. 1. Initially, the predicates that hold are $lose(0, \emptyset)$ and $win(\iota.0, \emptyset)$. Since $a \Leftarrow \iota.0$ and $b \Leftarrow 0$ we get $maze(M_{\texttt{ex}})(\emptyset) = \{a^+, b^-\}$. In this environment, $f \in lose(M_{\texttt{ex}}, \{a^+, b^-\})$ since $f \Leftarrow \iota.(\tau.a + \iota.\iota.\iota.0)$ and $win(\tau.a{+}\iota.\iota.\iota.0, \{a^+, b^-\})$, the latter essentially results from $win(\tau.a, \{a^+, b^-\})$. Next we derive $c \in win(M_{\texttt{ex}}, \{a^+, b^-\})$ from $c \Leftarrow \iota.(\iota.a + \iota.\iota.(\tau.b + \iota.\iota.0))$ and the fact that $win(a, \{a^+, b^-\})$ and $lose(b, \{a^+, b^-\})$ both hold. This shows $maze(M_{\texttt{ex}})^2(\emptyset) = \{a^+, c^+, b^-, f^-\}$. Another iteration confirms this as least fixed point, i.e., $\mu maze(M_{\texttt{ex}}) = \{a^+, c^+, b^-, f^-\}$.

Theorem 2.4 (Coincidence) *Let M be a maze and x be a variable.*

- *x is a winning position in M if and only if $x^+ \in \mu maze(M)$.*
- *x is a losing position in M if and only if $x^- \in \mu maze(M)$.*

The proof of this theorem can be adapted from results on finite symmetric and memory–free games [10]. The only slight twist is that our setting allows for two types of transitions in game graphs, namely visible and secret ones.

3 Esterel Reactions and Mazes

This section first formally presents the combinational fragment of Esterel which we are concerned with and recalls its constructive behavioral semantics as defined by Berry in [3]. We then give a translation of combinational Esterel programs P into mazes M with the property that winning (losing) positions in M correspond exactly to signals that must (cannot) be emitted in P.

Esterel Reactions.

The syntax of the Esterel fragment, which specifies single reactions and will be used in the remainder, is defined by the following BNF, where s stands for a signal name taken from some (finite) universe $\mathcal{S}$.

$P ::= 0$	`nothing`
$\mid\ !s$	`emit` s
$\mid\ s^+?(P)$	`present` s `then` P
$\mid\ s^-?(P)$	`present` s `else` P
$\mid\ P \mid P$	$P \,\vert\vert\, P$

Esterel's more general choice statement "`present` s `then` P_1 `else` P_2" can be recovered in our syntax by the term $s^+?(P_1) \mid s^-?(P_2)$ [6]. In this paper we omit the combinational operators for sequential composition and local signal declaration, though we discuss in Sec. 4 various ways for including the latter. A consequence of this omission is that the completion codes needed in the behavioral semantics' definition for the full language [3] become obsolete. We do away also with *input signals* $i \in I = \{i_1, \ldots, i_n\} \subseteq \mathcal{S}$. This is possible since the behavior of P under I is equivalent to the behavior of $P \,|\, !i_{j_1} \,|\, \cdots \,|\, !i_{j_m}$, where the indexes $j_1, \ldots, j_m \in \{1, \ldots, n\}$ are exactly those for which signal i_{j_k} is present in I [6]. Finally, we further assume that the finite set $\mathcal{S}$ of signals includes all signals of the combinational Esterel programs that one wishes to reason about. This restriction is a mere technical convenience and could be dropped by maintaining for every program the finite set of relevant signals, i.e., a "signal sort."

Berry's behavioral semantics for Esterel considers *statuses* of signals. Each signal $s \in \mathcal{S}$ can either be known to be present, i.e., have status s^+, or known to be absent, i.e., have status s^-, or have an unknown status. For any set $S \subseteq \mathcal{S}$ we let S^+ stand for $\{s^+ \,|\, s \in S\}$ and S^- for $\{s^- \,|\, s \in S\}$. *Consistent* sets $E \subseteq \mathcal{S}^+ \cup \mathcal{S}^-$ of signal statuses such that $\nexists s.\ s^+ \in E \textit{ and } s^- \in E$ are referred to as *events*, with $\mathcal{E}$ denoting the set of all events. Given a combinational Esterel program P and an event E, we define the sets $\mathit{must}(P, E)$ and $\mathit{cannot}(P, E)$ of those signals that must and cannot be emitted in P, respectively, relative to the knowledge of the signal statuses in E [3]:

$$
\begin{array}{lllll}
\mathit{must}(0, E) & := \emptyset & \mathit{must}(s^+?(P), E) & := & \begin{cases} \mathit{must}(P,E) & \text{if } s^+ \in E \\ \emptyset & \text{otherwise} \end{cases} \\
\mathit{must}(!s, E) & := \{s\} & \mathit{must}(s^-?(P), E) & := & \begin{cases} \mathit{must}(P,E) & \text{if } s^- \in E \\ \emptyset & \text{otherwise} \end{cases} \\
 & & \mathit{must}(P_1 \,|\, P_2, E) & := & \mathit{must}(P_1, E) \cup \mathit{must}(P_2, E) \\
\mathit{cannot}(0, E) & := \mathcal{S} & \mathit{cannot}(s^+?(P), E) & := & \begin{cases} \mathcal{S} & \text{if } s^- \in E \\ \mathit{cannot}(P,E) & \text{otherwise} \end{cases} \\
\mathit{cannot}(!s, E) & := \mathcal{S} \backslash \{s\} & \mathit{cannot}(s^-?(P), E) & := & \begin{cases} \mathcal{S} & \text{if } s^+ \in E \\ \mathit{cannot}(P,E) & \text{otherwise} \end{cases} \\
 & & \mathit{cannot}(P_1 \,|\, P_2, E) & := & \mathit{cannot}(P_1, E) \cap \mathit{cannot}(P_2, E)
\end{array}
$$

Note that both functions *must* and *cannot* are monotonic in E for subset inclusion. Since $(\mathcal{E}, \subseteq)$ is a finite $\cap$–semi–lattice, the function

$$\mathit{esterel}(P)(E) \ := \ \mathit{must}(P, E)^+ \cup \mathit{cannot}(P, E)^-$$

has a least fixed point $\mu\mathit{esterel}(P) = \bigcup_{i \in \mathbb{N}} \mathit{esterel}(P)^i(\emptyset)$. This least fixed point defines the behavioral semantics of P.

In [3] the construction of absent signals is based on the complement of the *cannot* sets, called *can* sets. Our equivalent formulation brings out an important structural invariant. Firstly, $\mathit{cannot}(P, E)$ is the logical dual of $\mathit{must}(P, E)$, obtained by interchanging $\cap$ for $\cup$, $\emptyset$ for $\mathcal{S}$, and $\{s\}$ for $\mathcal{S} \setminus \{s\}$. Secondly, *must* and *cannot* are exclusive: $\mathit{must}(P, E) \cap \mathit{cannot}(P, E) = \emptyset$, for any event E. In general, however, $\mathit{must}(P, E) \cup \mathit{cannot}(P, E) \neq \mathcal{S}$, whence *must* and *cannot* are not necessarily complements. This is analogous to the situation in two–player games: we will show below that *must* (*cannot*) corresponds to the construction of winning (losing) positions for the starting player. Elements neither in $\mathit{must}(P, E)$ nor in $\mathit{cannot}(P, E)$ indicate draw positions.

Representing Combinational Esterel Programs as Mazes.

With each Esterel program P we associate a maze $\langle\!\langle P \rangle\!\rangle := (a \Leftarrow \langle\!\langle P \rangle\!\rangle_a)_{a \in \mathcal{S}}$ that exactly reflects the semantics of P. Here, signals play the role of variables representing distinguished rooms, and the maze term $\langle\!\langle P \rangle\!\rangle_a$ describes the game conforming to P that can be played starting in room a. Formally, $\langle\!\langle P \rangle\!\rangle_a$ is defined along the structure of P as follows.

$$\langle\!\langle 0 \rangle\!\rangle_a := 0 \qquad \langle\!\langle s^+?(P) \rangle\!\rangle_a := \iota.(\iota.s + \iota.\langle\!\langle P \rangle\!\rangle_a)$$

$$\langle\!\langle !s \rangle\!\rangle_a := \begin{cases} \iota.0 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases} \qquad \begin{aligned} \langle\!\langle s^-?(P) \rangle\!\rangle_a &:= \iota.(\tau.s + \iota.\langle\!\langle P \rangle\!\rangle_a) \\ \langle\!\langle P_1 \,|\, P_2 \rangle\!\rangle_a &:= \langle\!\langle P_1 \rangle\!\rangle_a + \langle\!\langle P_2 \rangle\!\rangle_a \,. \end{aligned}$$

Intuitively, program 0 that cannot emit any signal, must correspond to a dungeon in which the leading player loses. The forced emission of signal a and absence of signals $s \neq a$ in program $!a$ leads to immediate success for the leading player in room $a \Leftarrow \iota.0$ —or unavoidable loss for the opponent— and loss in any other room $b \Leftarrow 0$. A program $s^+?(P)$ that is positively guarded by signal s must emit signal a if, in room a, the leading player has both a winning strategy for s *and* for room a in the maze corresponding to P. We encode this conjunction into mazes by first giving control to the opponent, who in turn freely decides whether the leading player must continue with his or her play in s *or* in $\langle\!\langle P \rangle\!\rangle_a$. Analogously, a program $s^-?(P)$ that is negatively guarded by signal s must emit a if, in room a, the opponent player has a winning strategy for s, *and* if the leading player has a winning strategy for $\langle\!\langle P \rangle\!\rangle_a$. Finally, an emission of signal a in a parallel composition may occur in either component and is thus encoded by a free choice for the leading player. In terms of maze graphs, parallel composition corresponds to forming the room–wise union (overlay) of corridors. As an example, the translation $\langle\!\langle P \rangle\!\rangle$ of the Esterel program $P_{\mathtt{ex}} = \,!a \,|\, e^+?(!d) \,|\, a^-?(!d \,|\, !f) \,|\, a^+?(b^-?(!c)) \,|\, g^+?(e^+?(g^-?(!e) \,|\, e^-?(!g)))$ (slightly optimised using the equivalence $\langle\!\langle Q \rangle\!\rangle_s \equiv 0$ when s does not occur in Q, as well as $m + 0 = m$, $\iota.\iota.m = m$ for all m) generates the unfolding rules of Example 2.1 and thus the maze in Fig. 1.

Proposition 3.1 *Let P be a combinational Esterel program and E an event.*

(i) $\mathit{must}(P, E) \quad = \{a \in \mathcal{S} \mid \mathit{win}(\langle\!\langle P \rangle\!\rangle_a, E)\}$

(ii) $\mathit{cannot}(P, E) = \{a \in \mathcal{S} \mid \mathit{lose}(\langle\!\langle P \rangle\!\rangle_a, E)\}$

This proposition states the desired one–to–one relation between the *must*– and *cannot*–analysis in combinational Esterel programs and the determination of winning and losing positions in their corresponding mazes. Its proof can be conducted by induction on the structure of Esterel programs. As a consequence of the proposition, the functions $\mathit{esterel}(P)$ and $\mathit{maze}(\langle\!\langle P \rangle\!\rangle)$ coincide. This immediately proves the following main theorem of this paper.

Theorem 3.2 (Game–Theoretic Characterization) *For every combinational Esterel program P, we have $\mu\mathit{esterel}(P) = \mu\mathit{maze}(\langle\!\langle P \rangle\!\rangle)$.*

We conclude this section by revisiting several of the pathological examples from Berry's book [3]. For each of these combinational Esterel programs P_i (indexed as in Berry's book), we take $\mathcal{S}$ to be the set of all signals occurring in P_i. The programs, except for P_6, are illustrated as mazes in Figure 2.

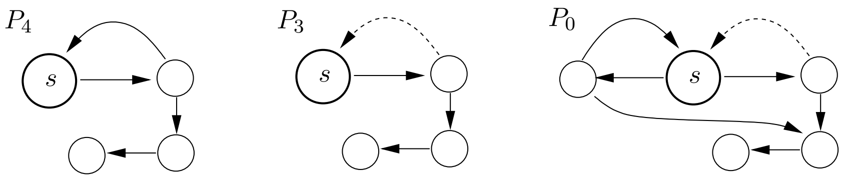

Fig. 2. “Pathological” Esterel examples.

- $P_4 := s^+?(!s)$: According to our definitions we obtain $M_4 := \langle\!\langle P_4 \rangle\!\rangle = (s \Leftarrow \langle\!\langle P_4 \rangle\!\rangle_s)$ and $m_{4,s} := \langle\!\langle P_4 \rangle\!\rangle_s = \iota.(\iota.s + \iota.\iota.0)$. Hence, for any event $E \in \mathcal{E}$:

$$\begin{aligned} \mathit{win}(m_{4,s}, E) &\iff \mathit{lose}(\iota.s + \iota.\iota.0, E) \\ &\iff \mathit{lose}(\iota.s, E) \text{ and } \mathit{lose}(\iota.\iota.0, E) \\ &\iff \mathit{win}(s, E) \text{ and } \mathit{lose}(0, E) \\ &\iff \mathit{win}(s, E) \text{ and } \mathit{true} \\ &\iff s^+ \in E\,. \end{aligned}$$

  Similarly, $\mathit{lose}(m_{4,s}, E) \iff s^- \in E$. This implies $\mathit{maze}(M_4)(\emptyset) = \emptyset$ according to the definition of function $\mathit{maze}(M_4)$, whence $\mu\mathit{maze}(M_4) = \emptyset$.

- $P_3 := s^-?(!s)$: Here we have $M_3 := \langle\!\langle P_3 \rangle\!\rangle = (s \Leftarrow \langle\!\langle P_3 \rangle\!\rangle_s)$ and $m_{3,s} := \langle\!\langle P_3 \rangle\!\rangle_s = \iota.(\tau.s + \iota.\iota.0)$. We may then derive, for any event $E \in \mathcal{E}$:

$$\begin{aligned} \mathit{win}(m_{3,s}, E) &\iff \mathit{lose}(\tau.s + \iota.\iota.0, E) \\ &\iff \mathit{lose}(\tau.s, E) \text{ and } \mathit{lose}(\iota.\iota.0, E) \\ &\iff \mathit{lose}(s, E) \text{ and } \mathit{true} \\ &\iff s^- \in E\,. \end{aligned}$$

  Moreover, $\mathit{lose}(m_{3,s}, E) \iff s^+ \in E$. As in the case for M_4, this implies $\mathit{maze}(M_3)(\emptyset) = \emptyset$ and thus $\mu\mathit{maze}(M_3) = \emptyset$.

- $P_0 := s^+?(!s) \mid s^-?(!s) = P_4 \mid P_3$**:** Hence, $M_0 := \langle\!\langle P_0 \rangle\!\rangle = (s \Leftarrow \langle\!\langle P_0 \rangle\!\rangle_s)$ and $m_{0,s} := \langle\!\langle P_0 \rangle\!\rangle_s = \langle\!\langle P_4 \rangle\!\rangle_s + \langle\!\langle P_3 \rangle\!\rangle_s$ according to the definition of $\langle\!\langle \cdot \rangle\!\rangle_s$. Given the calculations for P_4 and P_3 we easily conclude:
$$\begin{aligned} win(m_{0,s}, E) &\iff win(m_{4,s}, E) \ or \ win(m_{3,s}, E) \\ &\iff s^+ \in E \ or \ s^- \in E \\ &\iff lose(m_{0,s}, E)\,. \end{aligned}$$
It is easy to see that this yields again $\mu maze(M_0) = \emptyset$.
- $P_6 := s_0^+?(!s_1) \,|\, s_1^+?(!s_0)$**:** This combinational program gives rise to the maze $M_6 := \langle\!\langle P_6 \rangle\!\rangle = (s_0 \Leftarrow \langle\!\langle P_6 \rangle\!\rangle_{s_0}, s_1 \Leftarrow \langle\!\langle P_6 \rangle\!\rangle_{s_1})$, for $m_{6,s_0} := \langle\!\langle P_6 \rangle\!\rangle_{s_0} = \tau.\iota.(\iota.s_0 + \iota.0) + \tau.\iota.(\iota.s_1 + \iota.\iota.0)$ and $m_{6,s_1} := \langle\!\langle P_6 \rangle\!\rangle_{s_1} = \tau.\iota.(\iota.s_0 + \iota.\iota.0) + \tau.\iota.(\iota.s_1 + \iota.0)$. One can then easily infer, for $i \in \{1, 2\}$, that $win(m_{6,s_i}, E) \iff s_{1-i}^+ \in E$ and $lose(m_{6,s_i}, E) \iff s_{1-i}^- \in E$. Again, this implies $\mu maze(M_6) = \emptyset$.

In all of the examples above, the relevant rooms s, or s_0 and s_1 in case of M_6, are neither winning nor losing positions for the starting player but draw positions. According to Thm. 3.2, all signal statuses are undetermined, and our example programs would be rejected by current Esterel compilers [4]. The game metaphor, thus, yields a natural explanation of constructiveness and non–constructiveness.

4 Local Signal Declarations

This final section of our main development shows that the game semantics can also support *local signal declarations*, which is another kernel operation of Esterel [3]. A signal declaration $P\backslash s$ introduces a locally defined signal s that is available for broadcast inside program P only. As an example consider the program $(P\backslash s) \,|\, Q$ where $P := s^+?(!a) \,|\, !s$ and $Q := s^+?(!b)$. In Q, signal s refers to a different incarnation than the one used inside $P\backslash s$, whose scope is restricted by the local signal declaration. Consequently, the *internal* emission of s in $P\backslash s$ will trigger the emission of signal a in P but not that of signal b in Q.

The simplest way of interpreting local signal declarations in terms of mazes is to rename signal s in $\langle\!\langle P \rangle\!\rangle$ into some fresh signal name s', and to make sure that s' is never used outside of $\langle\!\langle P\backslash s \rangle\!\rangle$. While this "naming–apart" technique is a simple solution for a compiler, there is an algebraically more satisfactory way. Intuitively, $\langle\!\langle P\backslash s \rangle\!\rangle$ is the same as solving the recursive unfoldings characterizing maze $\langle\!\langle P \rangle\!\rangle$ with respect to signal s. If $s \Leftarrow m_s$ is the unfolding rule defining s, the least fixed–point solution for s is $\mu s.m_s$, where μ denotes the standard fixed–point operator known from process algebra. This recursive term for the game starting in room s is then used, or substituted, wherever s

is referenced in the unfoldings defined by $\langle\!\langle P \rangle\!\rangle$ for all the other rooms. Thus, s is eliminated. In our example, $\langle\!\langle P \rangle\!\rangle$ is the system

$$a \Leftarrow \iota.(\iota.s + \iota.\iota.0) + 0 \qquad b \Leftarrow \iota.(\iota.s + \iota.0) + 0 \qquad s \Leftarrow \iota.(\iota.s + \iota.0) + \iota.0\,,$$

from which we obtain the maze equations for $\langle\!\langle P \backslash s \rangle\!\rangle$ by substituting the fixed point $\mu s.\,(\iota.(\iota.s + \iota.0) + \iota.0)$ for every reference to s in the other expansions. The unfolding for s is "reset" to 0:

$$\begin{array}{ll} a \Leftarrow \iota.(\iota.(\mu s.\,(\iota.(\iota.s + \iota.0) + \iota.0)) + \iota.\iota.0) + 0 & \quad s \Leftarrow 0 \\ b \Leftarrow \iota.(\iota.(\mu s.\,(\iota.(\iota.s + \iota.0) + \iota.0)) + \iota.0) + 0\,. & \end{array}$$

In the new system of unfolding rules the behavior of the original signal s is completely encapsulated, so that any extension can only refer to the named rooms a and b. The signal name s remaining in the maze now refers to a fresh signal that is per default not emitted and hence represented by a dungeon. Obviously, the precise technical formalization of this idea requires the addition of the fixed–point operator $\mu s.m$ to our maze–term language.

Note that treating local signal declarations via fixed–point operators requires a slightly more general translation of $s^+?(P)$ and $s^-?(Q)$ than the one given above. This observation has already been made by Berry in [3]. Consider, e.g., a positive guard $c^+?(P \backslash s)$ which makes winning inside $P \backslash s$ dependent on winning signal c. This implies that any room in the maze of $P\backslash s$ can only be won under the extra condition that c can be won as well. This applies to all rooms inside $P \backslash s$, even to the "local" room s. To give the opponent a chance to force the starting player into room c at any point, we may break up each transition $P' \backslash s \xrightarrow{\iota} P'' \backslash s$ inside $P \backslash s$ via an auxiliary room in which the opponent can choose to challenge the starting player into room c or accept to continue with $P'' \backslash s$. Formally, we may replace $P' \backslash s \xrightarrow{\iota} P'' \backslash s$ by $P' \backslash s \xrightarrow{\iota} \iota.c + \tau.(P'' \backslash s)$, which can be done by appropriately modifying the operational rules for maze terms, especially the rules for *present* statements.

5 Conclusions and Future Work

This paper presented a game–theoretic semantics for combinational Esterel programs, i.e., for the kernel fragment of Esterel corresponding to combinational circuits. Our approach translated combinational programs into finite two–player games in such a way that Esterel's *must* and *cannot* analysis of signal statuses [3] could be rephrased as the computation of winning strategies. Our results complement the existing behavioral, operational, circuit–based, and model–theoretic approaches to Esterel's semantics [3,6]. In particular, they offer a novel didactic perspective for familiarizing students and engineers with this intricate constructive semantics, by giving emphasis to more intuition (game graphs) and less mathematics (fixed points). It is worth pointing out that our game–theoretic framework is not specific to Esterel but should apply to other synchronous languages incorporating the semantic principles of *synchrony* and *causality* as well, such as Pnueli and Shalev's Statecharts [9].

Regarding future work, we firstly want to extend our work to larger fragments of Esterel, especially those involving Esterel's *pause*, *wait*, and *loop* constructs [3]. This would allow for reasoning about sequences of reactions rather than single reactions, and requires enriching the information content held by states in our game graphs. It should also be investigated how our game semantics relates to the operational semantics implemented in state-of-the-art Esterel compilers. Secondly, we wish to study whether our approach is amenable to a rigorous algebraic treatment. Although we presented mazes in a process–algebraic fashion both in style of a term–based syntax and a transition–systems–based semantics, the associated algebraic theory has not yet been fully developed. An algebraic semantics for mazes could be based on a notion of behavioral equivalence, such as *bisimulation*. This would provide a *compositional* semantics for our games and allow for the minimization of mazes. Note that the translation from combinational Esterel programs into mazes presented in this paper does not apply any optimizations. Thirdly, we plan to develop a logical interpretation of our game semantics, similar to the one of AJM's games model in linear logic [2], thereby further exploring the logic behind Esterel [6]. As part of this plan, the relationship between our term language for mazes and Berry's calculus of *Constructive Boolean Logic* [3,11] deserves investigation.

References

[1] S. Abramsky. Games in the semantics of programming languages. In *11th Amsterdam Colloquium*, pages 1–5. Univ. of Amsterdam, 1997.

[2] P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Believe it or not, AJM's games model is a model of classical linear logic. In *LICS '97*, pages 68–75. IEEE Comp. Soc. Press, 1997.

[3] G. Berry. The constructive semantics of pure Esterel. CMA, Ecole des Mines, INRIA, 1999. Draft version 3.0.

[4] Esterel Technologies. Esterel Studio. www.esterel-technologies.com, 2003.

[5] M. Hyland and L. Ong. On full abstraction for PCF: I, II and III. *Inform. and Comput.*, 163(2):285–408, 2000.

[6] G. Lüttgen and M. Mendler. Towards a model-theory for Esterel. In *SLAP '02*, volume 65,5 of *ENTCS*. Elsevier Science, 2002.

[7] P. Malacaria and C. Hankin. Generalised flowcharts and games. In *ICALP '98*, volume 1443 of *LNCS*, pages 363–374. Springer-Verlag, 1998.

[8] A. Pietarinen and G. Sandu. Games in philosophical logic. *Nordic J. Philosophical Logic*, 4:143–173, 2000.

[9] A. Pnueli and M. Shalev. What is in a step: On the semantics of Statecharts. In *TACS '91*, volume 526 of *LNCS*, pages 244–264. Springer-Verlag, 1991.

[10] G. Schmidt and T. Ströhlein. On kernel of graphs and solutions of games: A synopsis based on relations and fixpoints. *SIAM J. Algebraic Discrete Methods*, 6:54–65, 1985.

[11] T. Shiple, G. Berry, and H. Touati. Constructive analysis of cyclic circuits. In *EDTC '96*, pages 328–333. IEEE Comp. Soc. Press, 1996.

[12] W. Thomas. On the synthesis of strategies in infinite games. In *STACS '95*, volume 900 of *LNCS*, pages 1–13. Springer-Verlag, 1995.