PUBLIC RESEARCH

KEYWORDS

process calculus, true concurrency, resource, consumption, recycling, quantification, pomset, denotational and structural operational semantics, full abstraction.

PUBLICATIONS (abstracts and introductions)

Fully Abstract Process Modelling over Recyclable Resources [10] (57 pages, full version of 9, to be submitted).

A Truly Concurrent Process Calculus over Recyclable Resources [9] (15 pages, submitted).

Abstract

In this paper we present a truly concurrent semantics for deterministic concurrent recursive processes accessing quantified recyclable resources. The process semantics is built upon the new coherently complete and prime algebraic domain of complex multi-pomsets, which is a quantitative version of the domain of complex resource pomsets. The CSP-like process language that we study contains several deterministic quantitative process operators, namely hiding, restriction, sequential, serial and parallel, as well as an observable recursion. A deterministic structural operational machine is displayed that allows extracting a linear and a complex operational semantics. The denotational semantics is naturally defined on \complex multi-pomsets for all finitary operators and lifted to a functional domain over environments modelling recursion. The robustness of the present semantical work is demonstrated by proving that the denotational semantics is fully abstract with respect to the linear and the \complex operational semantics and by relating it via observation to bisimilarity.

Introduction

Hennessy \& Plotkin~\cite{hp79} have shown in their seminal work how *power domains* can be employed in order to build semantic models of parallelism, by reducing it to interleaving and choice. In this paper we present a genuine approach to parallelism, that avoids using choice, thereby relying on *true concurrency*. The latter has motivated numerous domain-theoretic approaches based on variants of the event structures introduced by Winskel~\cite{win87}, of which the *pomsets* advocated by Pratt~\cite{pra86} are a particular but versatile case.

Along this line of research, an advanced truly concurrent approach, which has developed an operational and a matching denotational programming semantics modelling the customary process combinators excepting choice, was elaborated in Diekert \& Gastin~\cite{dg95icalp}, Gastin \& Teodosiu~\cite{gt02} and Gastin \& Mislove~\cite{gm02}. This work grounds upon the appealing interaction between processes and *resources* in any environment, a paradigm also driving the work in applied computer science and management. One should also note the related work of Pym \& Tofts~\cite{pt06} presenting a truly concurrent algebra and logic of processes and resources for a number of process combinators including choice.

A new stream in automata theory, as emphasized by the recent monograph on *weighted automata*~\cite{dkh09}, consists in attaching weights to transitions that measure their *cost* in terms of some available resources and extending these notions to the recognized words and languages. Classical

automata theory can in particular be recovered by considering unit weights, which is why the weighted view is more flexible, opening the way to new applications in engineering and economy.

A first attempt to develop a process language that combines the above truly concurrent approach on the denotational side with a weighted automata view on the operational side was undertaken in Teodosiu~\cite{teo12,teo12a} that operated an enrichment of the underlying domains and operators aimed at modelling processes accessing *consumable* resources. In that setting, resources reserved by a process are never returned to the environment upon termination, hence consumed. Therefore, the central idea there was to base the denotation on a resource semantics that *adds* the resources of sequentially composed processes. Most notably, hiding is a plus-action, hence non-idempotent, in this non-classical approach.

In this paper we deal with processes accessing *recyclable* resources. In this view, resources are allocated (reserved) by a process during execution and automatically freed (returned) to the environment upon termination. Therefore, the essential idea here is to base the denotation on a resource semantics that *joins* the resources of sequentially composed processes. We note that hiding is a join-action, hence idempotent, as in the classical approaches to process algebra.

We display a deterministic quantitative structural operational machine which engenders a linear operational semantics, that records only the strings of executed actions, and a complex operational semantics, that records the pomset of executed actions as well as the subsequent allocation of a process.

The denotational semantics is based on the domain of complex multi-pomsets. These are composed of an *observation*, which is a multi-pomset of presently executed actions, and of a *continuation*, which is a multi-set of subsequently allocated resources. The labeling of the denotational models and of the operational transition rules is quantified with the amount of resources being allocated. Quantification requires to replace the previously employed booleean algebra of sets of resources, with a *real algebra of multi-sets* of resources, while taking care that relevant algebraic properties remain valid in this new setting. Furthermore, it leads to defining operators whose quantitative semantics enriches that of previously considered operators.

The paper is organized as follows.

Section 2 recalls basic facts about partial orders.
Section 3 introduces the domain of \complex multi-pomsets.
Section 4 presents the process language and defines the allocation semantics.
Section 5 displays the deterministic structural operational machine, which engenders the linear and the complex operational semantics.
Section 6 is devoted to the ample compositional denotational semantics.
Section 7 presents the main results of full abstraction of the denotational semantics with respect to the linear and the complex operational semantics and relates it via observation to bisimilarity.

Fully Abstract Multi-Pomset Semantics for a Process Calculus over Consumable Resources. [8]
Theoretical Computer Science (TCS), Elsevier. (50 pages, full version of 7, submitted)

A Truly Concurrent Process Semantics over Multi-Pomsets of Consumable Resources [7]
Twenty-eighth Conference on the Mathematical Foundations of Programming Semantics MFPS XXVIII), Bath (UK), June 2012. (30 slides). Electronic Notes in Theoretical Computer Science (ENTCS) 286, pages 307-321, Elsevier, September 2012. (16 pages)

Abstract

This paper develops a truly concurrent semantical approach, whereby concurrency is notionally independent of nondeterminism, that allows describing the deterministically concurrent behaviour of recursive processes accessing consumable resources. The process semantics is based on the new coherently complete and prime algebraic domains of real and complex multi-pomsets. The process language that we study contains several deterministic quantitative process operators, namely a renaming, a hiding, a restriction, a serial and a parallel operator, as well as a recursion operator. The displayed deterministic structural operational machine engenders a
linear and a complex operational semantics. A compositional denotational semantics is constructed, which uses a functional domain over environments of complex multi-pomsets. The robustness of the presented semantical work is established by proving that the denotational semantics is fully abstract with respect to both linear and complex operational semantics.

Introduction

The seminal work of Hennessy \& Plotkin~\cite{hp79} has shown how *power domains* could be employed to build semantic models that support parallel composition of processes. Supposing that a choice operator is part of the language, parallelism is simply reduced to interleaving and choice, thus introducing nondeterminism into the semantic models.

In this paper, we follow a genuine approach to parallelism and concurrency, which avoids using choice and nondeterminism. We, thereby, rely on *true concurrency*, which has inspired a wealth of domain-theoretic approaches. These are mainly due to the insight that the prime *event structures* of Winskel~\cite{win87} are domains that provide suitable models to express truly concurrent process combinations.

However, to our knowledge, the only truly concurrent semantical approach that has actually developed an operational and a matching denotational programming semantics, allowing for finitary process combinators, as well as recursion, originates in the work of Diekert \& Gastin~\cite{dg95icalp}, Gastin \& Teodosiu~\cite{gt02} and Gastin \& Mislove~\cite{gm02}. The underlying denotational models are the *pomsets* advocated by Pratt~\cite{pra86} which, although being particular event structures, allow to simply express truly concurrent process combinations. Its intuition is based on the appealing interaction between processes and {\it resources} in any environment, a paradigm also driving the applied work in computer science of the last decades. One should also note the related work of Pym \& Tofts~\cite{pt06} presenting a truly concurrent algebra and logic of processes and resources for a number of process combinators including choice.

A new stream in automata theory, as emphasized by the recent monograph on *weighted automata*~\cite{dkh09}, consists in attaching weights to transitions which express their *cost* or *consumption* in terms of some available resources and to extend these notions to recognized words and languages. Classical automata theory can in particular be recovered by considering unit weights, which is why the weighted view is more versatile, opening the way to new applications in engineering and economy.

The present process language combines the above truly concurrent approach on the denotational side with a weighted automata view on the operational side, by enriching the underlying domains and operators in order to model processes {\it consuming} resources. To this end, the labeling of the denotational models and of the operational transition rules is quantified with the bounded or unbounded *time* or *amount* of resources being *consumed*.

Quantification requires to replace the previously employed *algebra of sets* of resources, seen as vectors over the booleans, by an *algebra of multi-sets* of resources, seen as vectors over the extended positive reals, while taking care that relevant algebraic properties remain valid in this new setting. Furthermore, it leads to defining operators whose quantitative semantics enriches that of previously considered operators (serial, parallel), together with operators having a new, quantitative semantics (renaming, hiding, restriction, recursion).

Our semantics differs from previous approaches in several technical respects. Firstly, the representation of processes as complex multi-pomsets relies on a real part and a *consumption part*, which allows to trivially check the consumption commutation and continuity. Secondly, the new *prefix partial order* on complex multi-pomsets substantially simplifies the proofs, allowing to check the complex continuity solely for the real part. Finally, the *semantics of recursion* relies on the continuity with respect to two orders instead of one.

The paper is structured as follows.

Section 2 recalls basic facts about {\it partial orders}.
Section 3 introduces the domains of {\it real and complex multi-pomsets}.
Section 4 presents the {\it process language} and defines the consumption semantics.
Section 5 displays a deterministic structural operational machine, which engenders the linear and complex {\it operational semantics}.
Section 6 is devoted to the compositional {\it denotational semantics} using a functional domain over environments of complex multi-pomsets.
Section 7 presents the main results of {\it congruence and full abstraction} of the denotational semantics with respect to both linear and complex operational semantics.

A Truly Concurrent Semantics for Processes sharing Quantified Resources [6]
PhD Thesis in Computer Science No.~2012PA077019, Department of Computer Science, University Paris 7, March 2012. (43 slides, 120 pages)

Abstract

In quest for a truly concurrent semantical approach, where concurrency is notionally independent of non-determinism, the paper presents the domain of finite and infinite complex multi-pomsets, which allow describing the deterministically concurrent behaviour of recursive processes.

The process language which we model is built on top of a fixed set of quantified resources that processes may access. It contains several finitary process operators, such as an empty process, action processes, a renaming, a hiding, a restriction, a sequential and an alphabetized parallel operator, as well as an infinitary recursion operator.

The defined structural operational machine specifies for each operator a small number of rewrite rules which engender the linear and complex structural operational semantics.

The denotation of a process is a complex multi-pomset that consists of two components: the first one is an already observed multi-pomset of quantified events, while the second one is a multi-set containing the quota of resources granted to the process for its execution. The defined approximation order confers to the set of complex multi-pomsets the structure of a coherently complete and algebraic Scott domain. The complex denotational semantics of the operators is then defined and shown to be Scott-continuous with respect to the approximation order.

The robustness of the presented semantical work is established by proving that the complex denotational semantics is adequate and fully abstract with respect to the linear and complex operational semantics.

Presentation

The main goal of process theory consists in lifting notions like concurrency, conflict, synchronization and communication, which are easily defined for atomic actions/events to the level of processes, viewed as compositions/aggregations of such.

Process Theory

Among the diverse approaches to process theory there are two notable mechanisms for describing the composition/aggregation of actions/events into processes.

Language Theory: the syntactical top-down view, a computational-operatorial approach, is that all processes decompose into atomic actions/events by a small number of operators (serial, parallel, choice, etc.) which instantiate the signature of an operator algebra of processes. Therefore, the set of all processes is described by an operatorial algebra, called a domain, which is a set acted upon by operators of a given, small signature. Central concerns are how to build reasonings about a process expression from reasonings about its parts and the way they compose, and how to relate different kinds of reasonings. In reasonings about concurrency one mostly adopts an interleaving stance equating parallelism with interleaving and choice, to the effect that parallelism inherently entails non-determinism, thus disregarding true concurrency.

Model Theory: the semantical bottom-up view, a representational-relational approach, is that each process is a relational structure on a set of actions/events that exist in a small number of relational bindings (serial, parallel, choice, etc.), which instantiate the signature of a relational algebra on action-events. Therefore, each process is described by a relational algebra, called a trace, which is a set of action-events linked through relations of a given, small signature. Central concerns are how to build the representation of a process from representations of its parts and the way they aggregate, and how to relate different kinds of representations. In representing concurrency one mostly adopts a non-interleaving stance distinguishing parallelism from interleaving and choice, to the effect that parallelism inherently reflects determinism, thus emphasizing true concurrency.

We would like to point out that, although the algebraic tools used by the first, operatorial approach have been more developed, due to the fact that the theory of universal operatorial algebra (Galois, Grassmann, Boole, Hamilton, Whitehead, Birkhoff, ...) had been extensively under study during the last century, the second, relational approach, has been meanwhile similarly developed into the theory of universal relational algebra (De Morgan, Pierce, Schr\"oder, Tarski, Moore, ...), albeit less known and practised than the former (\cite{bks97,mpm96}).

Essentially, from a purely formalistic perspective, it is a top-down theory of functions versus a bottom-up theory of relations. Only the later start and the lack of dissemination of the latter conveys the impression that in active research the relational point of view of process model theory lacks the formal tools for describing a theory of processes as clearly and cleanly as the operatorial point of view of process language theory.

In what follows, we first briefly present the two endeavours as applied to the main process language branches of CSP (see~\cite{hoa85,hj98,ros98}) and CCS (see~\cite{mil80,mil89}) and then aim at finding a truly concurrent combination between process language and model theory.

Process Language Theory

We start by sketching the language theoretic, operatorial approach.

Process language theory is less concerned about how processes are represented and in exchange stresses their interactional abilities. It assumes that processes interact through a small number of process operators of small arity (serial, parallel, choice, hiding, etc.), called process combinators. The expressions over constants and free variables using such combinators are generated by a context-free grammar and represent the terms of the process language.

Sometimes term grammars come along with a number of axiomatic equations or reductions over small language terms (associativities, neutral elements, comutativities, distributivities, etc.), which are intended to be satisfied by any denotation of the terms, and thereby constitute an axiomatic theory of processes. Some combinators and axioms are undisputable part of all axiomatic theories of processes, while others represent the particular bias of each process theory.

After several decades of diverse research in this area, there is presently a manifest concern in unifying the various process theories, by relating their expressiveness and complexity mutually and to the more classical theories of sequential computation (see~\cite{bae05,bb06,bbr10,par08}).

Process Languages

To make the picture more precise let us consider a minimal process term algebra built up from the set of actions $\Act$ and allowing for serial composition of processes, process variables from a set $\Var$ and recursive definitions, as expressed by the BNF-like grammar $p ::= a \mid p \serl p \mid x \mid \rec{x}{p}$, where $a \in \Act$ and $x \in \Var$ denote actions and variables, respectively.

A valid process term would be for instance $p = \rec{x}{((a \serl x) \serl y)}$, whose single free variable is y, the variable x being bound by recursion, and a being a constant.

Process terms are finitary if containing no recursion, such as $p = (a \serl x) \serl b$, and infinitary otherwise. Process terms are closed if containing no free variable, such as $p = \rec{x}{((a \serl x) \serl b)}$, and open otherwise.

Semantics

Competing as a formalization basis with the denotational is the operational semantics of process languages which, is occasionally preferred, because it reveals the underlying semantical intent more dynamically than the former.

Often, because of this circumstance, the operational description comes ahead of the denotational one, thus leading the search for suitable domains and models of the latter, which capture the essence of the particular notions involved.

Being a rather subtle and complex endeavour, this inquiry into true concurrency starts by laying down the dynamics, as expressed by the operational side, and follows up with a representation as expressed by denotational side.

Operational Semantics

A structural operational machine is given by a transition system $p \trans{a}{\tau} p'$ that defines the rewriting logic between terms p and p' of the process language and the associated observation a in an environment τ depending on the algebraic composition of the terms p and p'.

Such a rewrite rule might be

$$
\begin{array}{rlrl}
\displaystyle{
p \trans{a}{\tau} p'
\over
p \serl q \trans{a}{\tau} p' \serl q
}
\end{array}
$$

which, supposing a given observation a on p and an associated transformation to p', allows inferring the same observation a on the serial composition $p \serl q$ and an associated transformation to $p' \serl q$.

A single operator, such as the serial composition $\serl$, is subjected to several of the structural rewrite rules, thus generating an infinite transition system between process terms, labelled with the associated observations. The structural operational semantics of process terms is customary
defined by retaining the string of labels on chained rewrite transitions.

Denotational Semantic

A denotational semantics for this process term algebra requires to fix a ground domain $\D$ and to specify a denotation $\SEM{p}: \D^\Var \to \D$ for every term p of the language. Elements $\sigma \in \D^\Var = \Var \to \D$ which supply for every variable $x \in \Var$ a value $\sigma(x) \in \D$ from the ground domain are called environments. The semantics of a process term p is, therefore, a function that assigns to every environment $\sigma \in \D^\Var$ a value from the ground domain $\SEM{p}(\sigma) \in \D$.

Finitary Compositional Semantics

Choosing for instance for every action $a \in \Act$ a value $\SEM{a} \in \D$ and for the serial composition $\serl$ an operator $\SEM{\serl} : \D^2 \to \D$ allows one to define the semantics of composed process terms $p \serl q$ by $\SEM{p \serl q}(\sigma) = (\SEM{p}(\sigma))\SEM{\serl}(\SEM{q}(\sigma))$ and, subsequently, to extend it by structural induction to all finitary process terms, thus obtaining a finitary compositional semantics for all finitary terms of the process language.

Infinitary Recursion Semantics

The semantics $\SEM{\rec{x}{p}}(\sigma)$ of an infinitary recursive term $\rec{x}{p}$ in an environment σ is defined to be some fixed point of the associated evaluation mapping $\D \to \D, y \mapsto \SEM{p}(\sigma[x \mapsto y])$, where $\sigma[x \mapsto y]$ denotes the environment obtained from σ by assigning the value y to the variable x.

In order to make sure that such fixed points always exist and thereby be able to define the semantics of infinitary recursive terms, further structure of either order or metric nature has traditionally been imposed on the ground domain and the mappings involved, thus leading to two well-known, formally similar denotational approaches.

Order-Continous Semantics

In the historically first, order-based approach, originally laid out by D.~Scott for the purpose of modelling the λ-calculus, the ground domain $\D$ is endowed with the structure of a complete algebraic partial order also called a Scott domain. Choosing now $\SEM{\serl} : \D^2 \to \D$ to be Scott-continuous makes all finitary process terms define Scott-continuous evaluation mappings.

The theorem of Knaster, Tarski, Kleene and Scott stating that every Scott-continuous self-map of a Scott domain has a least fixed point and that this functional is in turn Scott-continuous, allows then defining the denotation of a recursive process term as the least fixed point of its evaluation mapping.

Metric-Contractive Semantics

The second, metric-based approach, which stemmed from classical functional analysis, consists in endowing the ground domain $\D$ with the structure of a complete metric space. Choosing now $\SEM{\serl} : \D^2 \to \D$ to be a contractive operator makes all guarded finitary process terms define contractive evaluation mappings.

Banach's fixed point theorem stating that every contractive self-map of a complete metric space has a unique fixed point, allows then defining the denotation of a guarded recursive process term as the unique fixed point of its evaluation mapping (see~\cite{bre98}). For a development using this approach see for instance~\cite{bw90}.

Enriched Categorial Semantics

This summary would not be complete without mentioning enriched categories as a unifying approach to both order and metric semantics, thereby allowing to define and compute fixed point solutions to recursions in a very abstract and universal framework (see~\cite{wag94}).

Because of the restraint in use of categorial language we have avoided for now inquiring into such a semantical approach to true concurrency that might do away with the numerous restrictions arising while taking the order or metric semantics as, at times manufactural and artificial, guiding line of discourse.

Congruence, Adequacy and Full Abstraction

The main purpose of defining a denotational and an operational semantics is to show that the two coincide according to some notion of observation equivalence extracted from the transition semantics, and thereby establish congruence, adequacy and full abstraction.

Process Model Theory

We now go over to sketching the model theoretic, relational approach.

True Concurrency

Historically to start with, the domain of terminated finite and infinite strings has been commonly used to define denotational semantics for process algebras. Within this well-known framework the denotational semantics of concurrency is derived via power domains from that of non-deterministic choice and interleaving to the effect that the denotational semantics of a concurrent process is equal to the set of all its possible finite and infinite sequential behaviours.

True concurrency marks a clear departure from this point of view by considering concurrency notionally independent of non-determinism. Stated plainly, concurrency is considered to be true, especially when deterministic.

The search for truly concurrent semantical approaches led to ever more general relational structures as denotational basis of process languages. In the next sections, we first briefly review the development of trace theory which has emphasized the truly concurrent approach, and then point out the difficulties that arise especially in dealing with recursion in the semantics of process languages.

Finitary Model Theory

Common to all models of true concurrency is the fact that, among the relations defined within a trace, there is one distinguished preorder relation reflecting serial occurrence of events, variously called precedence, causality or synchronization, which aims at specifying the temporal relation between the events of the trace.

More refined models of true concurrency usually assume the existence of further spatial, conflictual, etc. relations between the events of the trace modeling the parallel, choice, etc. occurrence of events.

Hoare Traces

Hoare traces are simply strings over an alphabet of actions. As such, they are sets of totally ordered events with a certain action-labelling.

Serial composition of Hoare traces is internal and simply modelled by catenation of strings. On the other hand, non-deterministic choice is not internal and is modeled through powersets of strings.

Most notably, in order to model deterministic parallel composition, one has to make use of interleaving and of the non-deterministic choice operator which is why the parallel composition is not internal on Hoare traces, as it renders the use of powersets necessary.

This enforces an interleaving point of view on the semantics of deterministic parallelism and prevents a truly concurrent approach. Stated differently, even deterministic parallelism introduces nondeterminism through the formalization bias, something which is viewed to be a theoretical artefact that does not correspond to the genuine descriptional intent.

Mazurkiewicz Traces

The theory of Mazurkiewicz traces has been used over the last decades as simple, but expressive tool, for investigating concurrent systems according to the non-interleaving viewpoint (overviews can be found in~\cite{maz87,ar88,per89,die90} and in the monograph~\cite{dr95}).

From an order-theoretic point of view, Mazurkiewicz traces can be regarded as being a special form of action-labelled partial orders on events which are ordered if and only if their labels are in conflict. The idea is to start with a finite alphabet $\Act$ of actions and an explicit symmetric and irreflexive

concurrency or independence relation specifying those pairs of actions that can execute concurrently. The complement relation is the conflict or dependence relation. The intended semantics is that performing two independent actions in any given order should lead to the same result.

Natural examples for modelling concurrency and conflict already arise when considering the simple case of actions sharing exclusive resources from some fixed set $\Res$. This can be expressed by a given resource map $\res : \Act \to \Pwset(\Res) \setminus \{\emptyset\}$ assigning to each action the non-empty set of resources it shares. Two actions are then considered to be concurrent if and only if they share no common resource, i.e. $a \disj b$ iff $\res(a) \Meet \res(b) \not= \emptyset$, conflicting otherwise. Moreover, any concurrency relation $\disj$ between actions can be induced on $\Act$, by choosing the set of conflicts as resource set $\Res = \bar{\disj}$ and by defining the resources of an action $a \in \Act$ to be precisely the incident conflicts $\res(a) = \{(b,c) \not\in \disj \mid b = a \txt{or} c = a \} \in \Pwset(\Res) \setminus \{\emptyset\}$.

Finite sequences of actions which can be mutually transformed into each other by commuting consecutive concurrent actions are considered to represent different sequential behaviours of one and the same concurrent process. Thus, finite concurrent processes are mathematically described as elements of the quotient monoid $(\M(\Act,\disj),\serl) = (\Act^*,\serl)/\SETOF{ab=ba}{(a,b) \in \disj}$, called the monoid of finite Mazurkiewicz traces over the independence alphabet $(\Act,\disj)$.

The catenation on finite Mazurkiewicz traces is inherited from $\Act^*$ and can be regarded as a serializable process composition: only dependent actions are synchronized, whereas at the process level no explicit synchronization is performed, thus retaining the maximal possible concurrency for the composed process.

This models in fact very well both extreme composition cases namely that of serial composition, in the string case $(\Act^*,\serl)$ of full dependency between all actions ($\disj = \Id_\Act$), and that of parallel composition, in the vector-addition case $(\N^\Act,+)$ of total independency between all actions ($\disj = \Act \times \Act$).

Thus partial serial and parallel compositions can be defined on Mazurkiewicz traces making use of the action labelling such that serial composition corresponds to total dependence and parallel composition corresponds to total independence between the labels of the two orders. Defining total and internal serial and parallel compositions for general dependence alphabets is nevertheless difficult to achieve. Moreover, more subtle operators, such as hiding, completely lack a clear interpretation over Mazurkiewicz traces.

This lack of modelling flexibility of Mazurkiewikz traces is the main reason why more general trace models are necessary in order to be able to accommodate the finitary semantics of process algebras.

Pratt Traces

A more versatile domain of traces from the point of view of the definable finitary process operators are the pomsets that have been advocated by Pratt as offering the natural model in order to express the serial and parallel compositions customary in process theory. A pomset is simply a labelled partial preorder of events whereby the partial order called precedence models the temporal ordering between events of the trace.

Similarly to Mazurkiewicz traces, Pratt's pomsets model very well both limit cases of composition, that of serial composition, in the string case, and that of parallel composition, in the vector-addition case, thereby offering a unifying language for generalizing various results previously established in either finite automata or Petri-net theory (see~\cite{pra86,bk91,kp94,bc87,gla96}).

Pomsets represent the theoretical basis that underlies the definition of Message Sequence Charts (MSC), which have been ever more employed and extended by industry as a standardized truly concurrent model able to express and allowing to process communication that is distributed and causality based.

This flexible and adaptive nature of pomsets accounts for their main attractiveness as a model for concurrency and has led over the last decades to increasing interest in using pomsets as a denotational domain for defining truly concurrent semantics of process algebras, a framework which we too adopt as the modeling basis of the semantical work that we shall present.

Winskel Traces

Winskel traces are event structures defined as being sets of event configurations together with an enabling relation between configurations and events that preserves lack of conflict. Winskel traces are very flexible models of concurrency, conflict and non-determinism and therefore of true concurrency (see~\cite{win87,wi09a}).

We shall not delve into the vast subjects and literature pertaining to event structures and rather restrict in what follows to the simpler but expressive enough case of pomsets.

Infinitary Model Theory

All trace models that have been mentioned so far account for suitable compositional interpretations of some of the finitary operators of process languages with increasing relevance of true concurrency.

Recursion

However, apart from the finitary algebraic structure modelling the process operators compositionally, additional infinitary properties must be satisfied by a denotational semantics in order to be able to define a rich language semantics. If recursion is to be allowed as an operator of the process language, the main concern is that of assuring the existence of fixed points for recursively defined terms.

Unfortunately, neither of the two sketched order/metric denotational frameworks for defining and computing fixed points of recursion readily applies to finite traces and the defined finitary operations. The main difficulty resides in defining an order/metric structure on the set of finite and infinite traces such that the operations be order-continuous/metric-contractive. Surmounting this obstacle called for a number of gradual adjustments, that apply equally well to the trace models of Hoare, Mazurkiewicz, Pratt and Winskel.

Real Traces

A natural partial order on finite traces, whether those of Hoare, Mazurkiewicz, Pratt or Winskel, generalizing the classical one used for strings is the prefix order. As in the well-known string case, the prefix order structure on finite traces $(\F,\leq)$ is algebraic, but not complete and, likewise, the natural prefix ultrametric structure on finite traces $(\F,d_{\rm pref})$ fails to be complete.

Applying the standard ideal completion device to $(\F,\leq)$ leads to the Scott-domain of real traces denoted by $(\R,\leq)$ (\cite{gas90,gr93,gp95}). Similarly, the standard Cauchy-completion of $(\F,d_{\rm pref})$ leads to a complete prefix ultrametric structure $(\R,d_{\rm pref})$ on real traces (\cite{kwi90}). Unfortunately, though having the right order and metric structure, the set of real traces

lacks the essential algebraic structure, since the operations extending those for finite traces can only be partially defined.

In addition to all these difficulties in defining a trace domain having convenient order/metric and algebraic structures comes the fact that in all cases the operations are not even monotone, let alone Scott-continuous, with respect to the prefix order. This comes indeed as no surprise, since it is already the case for the classical domain of finite strings $\Act^*$, as illustrated by the simple example of serial composition $\epsilon \leq a$, $b \leq b$, but $\epsilon \serl b \not\leq a \serl b$.

Complex Traces

The, by now, classical solution employed for inducing a Scott-domain structure on strings and to render string operations Scott-continuous is to go over to the domain of terminated finite and infinite strings $\Act^\infty\bot \Join \Act^*\top$, with $\Act^\infty=\Act^* \Join \Act^\omega$, whereby the trailing symbols $\top$ and $\bot$ signal string termination and non-termination, respectively.

This technique of explicit termination has been extended to the setting of Mazurkiewicz traces in~\cite{die93concat,die93mfcs,gp92mfcs,dg95icalp,teo93}, where the domains of complex α- and δ-traces have been defined and studied. The order and algebraic structure of real traces $(\R,\leq)$ can thus be reconciled by considering the ordered algebra of complex traces $(\C,\aleq)$. Moreover, a natural prefix-based ultrametric structure $(\C,d_{\rm pref})$ can be defined on complex traces, such that the operations be uniformly continuous yet non-contractive, which is why it is preferable to use theorder-theoretic rather than the metric-theoretic framework for defining the semantics of recursion.

Our Truly Concurrent Process Semantics

Turning now over to commenting our own approach, we first state its purpose, next sketch its content, then relate it to previous approaches and finally present its plan.

The Purpose of the Present Approach

We aim to model a process language which contains several finitary process operators, as well as an infinitary recursion operator, which enable us to expresstypical truly concurrent process combinations. We should exhibit a structural operational semantics that is straightforward to comprehend allowing to extract deterministic linear and complex behavior out of the process terms of our language. We should also present a denotational semantics which makes use of a trace ground domain that is algebraic and complete allowing to denote the finitary process operators as well as the infinitary recursion operator by suitable continuous interpretations. By defining a denotational and an operational semantics the main objective pursued resides in showing that the two match according to congruence, adequacy and full abstraction results established to this purpose.

A Sketch of the Present Approach

The domain of complex multi-pomsets proposed in this paper follows the general line of thought developed in previous approaches to truly concurrent finitary semantics over traces (Hoare, Mazurkiewicz, Pratt, Winskel) and, furthermore, extends the semantics to the infinitary recursive part of the process language.

Aiming at a truly concurrent semantical approach, we show that the domain of finite and infinite complex multi-pomsets is a suitable denotational domain of process semantics that capture the deterministically

concurrent behavior of recursive processes thereby extending and refining previous work of Diekert, Gastin, Teodosiu and Mislove~\cite{dg95icalp,gt02,gm02} on similar semantic domains designed for this purpose.

The intuition that underlies the denotational and operational semantics of the language operators is easy to understand being naturally based on the appealing interaction between processes and resources in any environment, a paradigm also driving the applied work in computer science of the last decades.

The Process Language

The process language which we model is built on top of a fixed set of quantified resources that processes may access. It contains several finitary process operators, such as an empty process, action processes, a renaming, a hiding, a restriction, a sequential an alphabetized parallel operator, as well as an infinitary recursion operator.

We also consider derived operators, notably, a sequential and a semi-dependent serial composition (both derived from the plain serial composition), and furthermore a partitional composition (derived from the alphabetized parallel composition).

The finitary operations which we define are internal and have the expected axiomatic properties (associativity, commutativity, neutral element, distributivity), thus inducing the desired algebraic structure.

The Operational Semantics

The defined structural operational machine specifies for each operator a small number of rewrite rules between process terms. Every rewrite rule supposes given observations and associated transformations of the term factors and provides an observation and an associated transformation of the term itself in a given resource environment. The application of these transition rules by structural induction to process terms engenders, as usually, the structural operational semantics.

We also consider the operational semantics of the derived operators and show how they relate to the finitary operators of our language.

The Denotational Semantics

The domain of complex multi-pomsets which we present extends the central algebraic properties of the domain of terminated finite and infinite strings. The body of properties thereby established allows devising a truly concurrent denotational semantics for our process algebra whose intuition is based upon the interaction between processes and resources in any environment.

The denotation of a process is a complex multi-pomset that consists of two components: the first one, the real part, is an already observed multi-pomset of events, the second one, the resource part, is a multi-set containing the quantity of resources granted to the process for its execution.

This approach is in fact similar to the failure (or ready) semantics used for process algebras like CSP and CCS~\cite{hoa85,mil80}. The difference is that whereas in failure semantics the quota restriction required by the resource part applies merely to the next process step, in the present model it applies to the whole process.

The labelling of the multi-pomset assigns to each event the quantity of resources it accesses, while the partial order of the multi-pomset expresses the temporal order between the events of the process. Since access to resources is supposed to be mutually exclusive, those events which access common resources are considered to be unable to execute simultaneously and, therefore, required to be temporally ordered. This assumption also means that the temporal order on events is over-synchronized with respect to the event conflicts due to competing access to resources.

Specification refinement leads to a natural approximation order between complex multi-pomsets. It confers to the set of complex multi-pomsets the structure of a coherently complete and prime algebraic Scott domain. The denotation of the finitary operators is then defined and shown to be Scott-continuous with respect to the approximation order. This allows extending the denotational semantics to the whole process language, by defining the semantics of recursive terms making use of a least fixed point approach adapted to resource environments appropriately.

We also mention the fact that an ultrametric distance based on the approximation order can be exhibited which turns the set of complex multi-pomsets into a compact (hence complete) separable ultrametric space. The topology induced by the ultrametric coincides with the compact Lawson topology~\cite{law88} associated with the approximation order, allowing to easily transfer convergence results to-and-fro between the order and metric structures. Moreover, the operators are uniformly continuous, but not contractive with respect to the defined ultrametric, which is why we base our semantics on the order rather than on the metric perspective.

Congruence, Adequacy and Full Abstraction

The robustness of the presented semantical work is established by proving congruence, adequacy and full abstraction results between the denotational and operational semantics thus providing the suitable link in order to transfer semantical properties back and forth between the two frameworks.

Most notably, we go beyond defining the customary linear behaviour of concurrent processes and extract complex behaviour from the operational transition system which, thereby, reveals an operational approach that matches the presented denotational approach to truly concurrent process semantics.

Relation to Previous Approaches

We generalize the approach put forward in the work of Diekert \& Gastin~\cite{dg95icalp}, Gastin \& Teodosiu ~\cite{gt02} and Gastin \& Mislove ~\cite{gm02} by enriching the underlying models and operators in order to treat the case of processes using quantified resources. To this end, the ground domain of pomsets is generalized to the effect that the labeling of pomsets is quantified to reflect the bounded or unbounded amount of resources being consumed by each event.

The quantitative treatment of resources requires to exchange the previously employed algebra of sets seen as vectors over the booleans with an algebra of multi-sets seen as vectors over the extended natural numbers. In particular it makes necessary to replace the customary set-theoretic operators by the more general multi-set-theoretic operators on vectors, while taking care that the relevant algebraic properties remain valid under this extension.

Quantification leads us to defining operators whose semantics enriches that of the previously considered operators (serial, alphabetized parallel, recursion) together with operators having a completely new semantics (renaming, hiding, restriction), as follows.

The new quantified **renaming** operator allows substituting namespaces of resources. The semantics of this operator is straightforwardly defined to match the quantitative approach.

A new quantified **hiding** operator is defined, which is indexed by a quota of resources to be hidden. This operator erases a given consumption out of the observable effect of each process by internalizing the respective consumption. The semantics of this quantified version of the operator has been revised and genuinely extended to match the quantitative approach.

An equally new quantified **restriction** operator is defined, which is indexed by a quota of resources to be respected. This operator confines processes to the given amounts of consumed resources, thus preventing the execution of events that trespass the specified quota bounds. The semantics of this operator has been worked out to match the quantitative approach.

The **serial** composition is extended such that it applies to quantified events. While the operational semantics largely remains unchanged, the denotational semantics has been enriched for this quantified version. The serial composition is used in order to derive the **semi-dependent serial** composition obtained from the plain serial composition by renaming, which allows one to easily express synchronizations typical of semi-trace theory.

As a particular case of the serial composition we obtain the derived **sequential** composition, which models the complete serial synchronization of events of the compounds. We note that, contrary to the serial composition, the derived sequential composition has a simple denotational semantics, which is why we chose to present its semantics separately.

The **alphabetized parallel** composition is equally extended such that it applies to quantified events. While the operational semantics largely remains unchanged, the denotational semantics has been equally enriched for this quantified version.

As a particular case of the alphabetized parallel composition we obtain the derived **partitional** composition, which models the totally unsynchronized parallel composition of events of the compounds. We note that, contrary to the alphabetized parallel composition, the derived partitional composition has a simple denotational semantics, which is why we chose to present its semantics separately.

The **recursion** operator is extended in order to take into account the consumption of resources as expressed by the quantitative resource semantics. The semantics of this operator has been considerably extended in order to treat the quantitative approach.

The denotational semantics of our process language differs from previous approaches in several crucial respects.

First, the representation of processes as complex multi-pomsets no longer relies on a real and an imaginary part, but on a **real** and a **resource part**. This new representation of the complex semantics greatly simplifies the presentation since the resource part directly reflects the trivial resource consumption semantics, which is no more just derivable from the complex semantics, but actually a component. In the quantitative approach that we present the imaginary part of a process may be straightforwardly defined from the real and resource parts, but is employed only for defining the complex semantics of serial compositions, while remaining largely irrelevant in the definition of the complex semantics of all other operators. This representation also allows one to trivially check the functional conditions of commutation and continuity pertaining to resources for all finitary operators.

Secondly, the new **approximation partial order** on complex multi-pomsets has been substantially simplified such that the real component reflects prefixing while leaving the resource component completely unchanged. With respect to this new approximation ordering directed subsets of processes preserve a constant resource part, therefore, the functional condition of complex continuity practically has to be solely checked for the real part. Previous approaches essentially relied on residuals for defining the approximation relation on the imaginary parts and were much more difficult to comprehend and handle, in particular in the proofs of complex continuity of the operators.

Finally, the **semantics of recursion** makes use of three functional conditions, resource commutation, resource continuity and complex continuity that it preserves on application and which the denotation of all finitary operators are shown to fulfil. In this respect, the denotation of recursion is more involved than in previous approaches since it relies on the continuity with respect to two orders instead of one. This enables us to define and compute least fixed points of term semantics with respect to both orders, first in the domain of resources and subsequently in the domain of complex multi-pomsets in a two stage least fixed point application process.

The Plan of the Thesis

After this introduction, the presented material is structured in five chapters as follows.

Chapter 2 recalls the basic facts about {\bf partial orders} fixing the pertinent notions and notations.
Chapter 3 presents the {\bf process language}. We discuss its grammar, its signature, and, most notably, present a trivial quantitative resource semantics that is central to the subsequent denotational and operational semantics.
Chapter 4 presents the {\bf operational semantics}. We first display a structural operational machine for our language revealing the dynamic intuition behind the language operators. We then present some notable examples of derived operators of our language and their respective operational semantics.
Chapter 5 presents the {\bf denotational semantics}. We proceed in two stages. First, we define the domain of multi-pomsets and present a compositional denotational semantics for all finitary operators of the language (Section 5.1). We then note that the domain of multi-pomsets is unsuited in order to interpret the remaining infinitary recursion operator due to the lack of compatibility between the defined order and finitary operations. Secondly, we define the domain of complex multi-pomsets and present a compositional and continuous denotational semantics for all finitary operators of the language, as well as for the infinitary recursion operator (Section 5.2). We show that the defined order and finitary operations are compatible and allow interpreting the infinitary recursion operator using an adapted least fixed point approach whose intuition is centered upon resources.
Chapter 6 presents the results of {\bf congruence, adequacy and full abstraction}. First we work out a simple linear translation between linear prefixing on the denotational side and linear transition on the operational side (Section 6.1). Next, we define the linear operational behaviour of language term and prove a linear congruence result which allows us to infer the linear adequacy and full abstraction of the denotational with respect to the operational semantics (Section 6.2). Finally, we define the complex behaviour of language terms making use of the well-known ability of multi-pomsets to be linearized, and conclude with complex congruence, adequacy and full abstraction results proving the desired match between the denotational and operational semantics (Section 6.3).

We end the exposition with a conclusion containing a summary, some alternatives and an outlook.

Resource Traces: A Domain for Processes Sharing Exclusive Resources [5]
Invited talk, Twelfth Conference on the Mathematical Foundations of Programming Semantics (MFPS XII), University of Colorado, Boulder (USA), June 1996. Theoretical Computer Science (TCS) 278,

pages 195-221, Elsevier, May 2002. Allocated research, Department of Computer Science, Université Paris 6, 1994-1997. (25 pages)

Abstract

The domain of explicitly terminated finite and infinite words is commonly used to define denotational semantics for process algebras such as CSP. In this well-known framework the denotational semantics of concurrency is derived via power-domains from that of non-deterministic choice and interleaving to the effect that the denotational semantics of a concurrent process is equal to the set of all its possible finite and infinite sequential behaviours.

In this paper we define a more versatile domain of so called finite and infinite resource traces which allow to capture the concurrent behaviour of a process and encode the static concurrency of a system directly into the domains definition. The approach we present refines previous work of Diekert (1990) and Diekert & Gastin (1993) on α- and δ-traces.

We start with an alphabet of atomic actions, a set of resources, and a resource map assigning to each action the non-empty subset of resources it uses. Actions that do not share common resources are called independent and considered to be able to execute concurrently. A partially terminated concurrent process is specified by a resource trace which consists of two components: an already observed part represented as an action-labeled partial order (Mazurkiewicz trace), and a guard set containing the resources granted to the process for its further development. A process concatenation is then defined which allows independent actions to execute concurrently.

Specification refinement leads to a natural approximation ordering between processes. It confers to the set of all processes the structure of a coherently complete prime algebraic Scott domain whereby process concatenation is Scott-continuous in both arguments.
Furthermore, we define a natural ultrametric on processes based on prefix information. he induced topology is shown to be equivalent to the compact Lawson topology induced by the approximation ordering. Process concatenation is moreover shown to be uniformly continuous with respect to the defined ultrametric.

The mathematical theory we develop thus extends the central order and metric properties of the domain of explicitly terminated finite and infinite words which are needed in order to devise truly concurrent semantics for process algebras much in the style of classical CSP semantics.

Introduction

The theory of Mazurkiewicz traces has been recognized over the last decade as an expressive tool for investigating concurrent systems according to the non-interleaving viewpoint (overviews can be found in~\cite{maz87,ar88,per89,die90} and in the recent monograph~\cite{dr95}).

The idea is to start with a finite alphabet $\S$ of actions and an explicit symmetric and irreflexive independence relation $\I \subseteq \S \times \S$ specifying those pairs of actions that can execute concurrently. The complement $\D = \S \times \S \setminus \I$ is called dependence relation. The intended semantics is that performing two independent actions in any order should lead to the same result.

Natural opportunities for defining dependence alphabets arise when considering actions that share exclusive resources from some fixed set $\Res$. This can be expressed by a resource map $\res : \S \to \Pwset(\Res) \setminus \{\emptyset\}$ assigning to each action the non-empty set of resources it uses. An induced dependence relation is then implicitly defined by setting two actions dependent iff they share some common resource, i.e. $a \Drel b$ iff $\res(a) \Intsec \res(b) \not= \emptyset$. Moreover, any dependence relation $\D$ can be induced on Σ by choosing the set of dependencies as resource set $\Res = \D$ and by defining the resources of an action $a \in \S$ to be precisely the incident dependencie $\res(a) = \{(b,c) \in \D \mid b = a \txt{or} c = a \} \in \Pwset(\Res) \setminus \{\emptyset\}$.

Finite sequences of actions which can be mutually transformed into each other by commuting consecutive independent actions are considered to represent different sequential behaviours of one and the same concurrent process. Thus, finite concurrent processes are mathematically described as elements of the

quotient monoid $(\M\SD,\cdot) = (\S^*,\cdot)/\SETOF{ab=ba}{(a,b) \in \I}$, called the monoid of finite Mazurkiewicz traces over the dependence alphabet $\SD$. The concatenation operation on finite traces is inherited from $\S^*$ and can be regarded as a weak sequential process composition: only dependent actions are synchronized whereas at the process level no explicit synchronization is performed, thus retaining the maximal possible concurrency for the composed process. It models in fact very well both extreme cases namely that of sequential composition, in the word case $(\S^*,\cdot)$ of full dependency between all actions ($\D = \S \times \S$), and that of parallel composition, in the vector-addition case $(\S^\N,+)$ of total independency between all actions ($\I = \S \times \S \setminus \Id_\S$), thereby offering a unifying language for generalizing various results previously established in either finite automata or Petri-net theory.

This versatile nature of trace theory accounts for its main attractiveness as a model for concurrency and has led over the last decade to increasing interest in using traces as a denotational domain for defining truly concurrent semantics of CSP-similar process algebras. However, apart from the monoidal structure modeling process concatenation, additional properties must be satisfied by a denotational domain in order to be able to complete such a task.

To make the picture more precise let us consider a minimal process term algebra build up from the set of actions $\S$ and allowing for concatenation of processes, process variables from a set V and recursive definitions, as expressed by the BNF-like syntax $p ::= a \mid p \cdot p \mid x \mid \rec x.p$ where $a \in \S$ and $x \in V$ denote actions and variables respectively. A valid process term would be for instance $p = \rec x.((a \cdot x) \cdot y)$ whose single free variable is y. Finitary process terms are those which contain no recursion such as $p = (a \cdot x) \cdot b$. Closed process terms are those which contain no free variables such as $p = \rec x.((a \cdot x) \cdot b)$.

A denotational semantics for this process term algebra requires to fix a ground domain $\Dom$ and to specify a denotation $\sem{p}: \Dom^V \to \Dom$ for every term p of the language. Elements $\sigma \in \Dom^V = V \to \Dom$ which supply for every variable $x \in V$ a value $\sigma(x) \in \Dom$ from the ground domain are called environments. The semantics of a process term p is therefore a function which to every environment $\sigma \in \Dom^V$ assigns a value from the ground domain $\sem{p}(\sigma) \in \Dom$. Choosing for instance for every $a \in \Sigma$ a value $\sem{a} \in \Dom$ and an operation $\sem{\cdot} : \Dom^2 \to \Dom$ allows to define the semantics of composed process terms $p \cdot q$ by $\sem{p \cdot q}(\sigma) = (\sem{p}(\sigma))\sem{\cdot}(\sem{q}(\sigma))$ and subsequently extend it by structural induction to all finitary process terms. The semantics $\sem{\rec x.p}(\sigma)$ of a recursive term $\rec x.p$ in an environment σ is then defined to be some fixed point of the associated evaluation mapping $\Dom \to \Dom, y \mapsto \sem{p}(\sigma[x \mapsto y])$, where $\sigma[x \mapsto y]$ denotes the environment obtained from σ by changing its value in x to y. In order to make sure that such fixed points always exist further structure of either order or metric nature has to be imposed on the ground domain and the mappings involved leading to two well-known formally similar denotational approaches.

In the historically first, order-based approach, originally laid out by D. Scott for the purpose of modeling the λ-calculus, the ground domain $\Dom$ is endowed with the structure of a complete algebraic partial order also called a Scott domain. Choosing $\sem{\cdot} : \Dom^2 \to \Dom$ to be a Scott-continuous operator makes all finitary process terms define Scott-continuous evaluation mappings. The theorem of Tarski, Knaster and Scott stating that every Scott-continuous self-map of a Scott domain has a least fixed point and that this functional is in turn Scott-continuous, allows then to define the denotation of a recursive process term as the least fixed point of its evaluation mapping.

The second, metric-based approach, which stemmed from classical functional analysis, consists in endowing the ground domain with the structure of a complete metric space. Choosing now $\sem{\cdot} : \Dom^2 \to \Dom$ to be a contractive operator makes all guarded finitary process terms define contractive evaluation mappings. Banach's fixed point theorem stating that every contractive self-map of a complete metric space has a unique fixed point, and that this functional is in turn contractive, allows then to define the denotation of a guarded recursive process term as the unique fixed point of its evaluation mapping.

Unfortunately, neither of the two denotational frameworks just sketched readily applies to finite traces and their concatenation defined above. The main difficulty resides in defining an order/metric structure on the set of finite traces such that the concatenation operation be Scott-continuous/contractive. Surmounting this obstacle called for a number of gradual adjustments of the original trace model, which we next briefly

review.

A natural partial order on finite traces over $\SD$ generalizing the one used for words is the prefix ordering derived from the concatenation operation which is defined for all $x,y \in \M\SD$ by $x \leq y$ iff there exists $z \in \M\SD$ such that $x \cdot z = y$. As in the well-known word case, the prefix order structure $(\M\SD,\leq)$ is algebraic but not complete and, likewise, the natural prefix ultrametric structure $(\M\SD,d_{\rm pref})$ fails to be complete. Applying the standard ideal completion device to $(\M\SD,\leq)$ leads to the Scott-domain of real traces over $\SD$ denoted by $(\R\SD,\leq)$. Similarly, the standard Cauchy-completion of $(\M\SD,d_{\rm pref})$ leads to a complete prefix ultrametric structure $(\R\SD,d_{\rm pref})$ on real traces. Unfortunately, though having the right order and metric structure, the set of real traces lacks the essential monoidal structure since a concatenation operation extending that for finite traces can only be partially defined. The monoidal structure can be mended by considering the monoid of complex traces \cite{die93concat}, but then, in exchange, the order structure defined by the prefix relation, while being complete, is no more algebraic \cite{gp92mfcs,die93mfcs,teo93}. Moreover, a natural prefix-based ultrametric structure $(\C\SD,d_{\rm pref})$ can be defined on complex traces, such that the concatenation operation is uniformly continuous, yet, non-contractive.

In addition to all these difficulties in defining a trace domain having convenient order and monoid structures comes the fact that in all cases the concatenation operation is not even monotone, let alone Scott-continuous, with respect to the prefix ordering. This comes indeed as no surprise, since it is already the case for the classical domain of finite words $\S^*$, as illustrated by the simple example $\epsilon \leq a$, $b \leq b$ but $\epsilon \cdot b = b \not\leq ab = a \cdot b$. The, by now, classical solution employed in order to induce a Scott-domain structure on words and to render word concatenation Scott-continuous is to go over to the domain of explicitly terminated finite and infinite words $\S^\infty\bot \Union \S^*\top$, whereby the trailing symbols $\top$ and $\bot$ respectively signal word termination and non-termination.

This technique of explicit termination has been extended to the setting of trace theory in~\cite{die93mfcs,dg95icalp}, where the domains of α- and δ-traces have been defined. Since our aim is to describe concurrent systems, we are not only concerned with full termination but also with partial termination. A partiallyterminated trace consist of an already observed trace and some alphabetical restriction on the continuation, which is operationally similar to the above termination symbols and guarantees the Scott-continuity of the defined concatenation operation. The approach is in fact similar to the failure (or ready) semantics used for process algebras like CCS and CSP~\cite{hoa85,mil80}. The difference is that whereas in failure semantics the alphabetical restriction applies merely to the next process step, in the trace models it persists on all future steps of the process.

The domain of resource traces proposed in this paper follows the general line of thought developed in~\cite{dg95icalp}. However, it uses in an essential way the representation of dependence alphabets in terms of resource mappings in order to define appropriate monoidal and order/metric structures on the set of resource traces. The concatenation operation we define is associative and has a neutral element thus conferring to the set of resource traces the desired monoidal structure. Also, an approximation ordering is defined which exhibits on the set of traces the structure of a coherently complete prime algebraic Scott-domain. The concatenation operation is then proved to be Scott-continuous with respect to the order structure. Furthermore, an ultrametric distance is defined which turns the set of traces into a compact (hence complete) separable ultrametric space. The concatenation operation is subsequently proved to be uniformly continuous with respect to the ultrametric structure. In addition, the topology induced by the ultrametric is shown to coincide with the compact Lawson topology~\cite{law88} associated with the approximation ordering, which allows to easily transfer convergence results between the order and metric structures.

The body of order and metric properties thus established for the domain of resource traces opens the way to devising truly concurrent semantics for processes algebras similar in style and spirit to CSP. A full development of such an approach to true concurrency, containing a denotational and a matching operational process semantics, will be presented in the forthcoming paper~\cite{gastmis99}.

We use a rich mathematical arsenal comprising techniques from algebra, domain theory and topology in order to present a generalization of the domain of explicitly terminated finite and infinite words which promises to be adequate from an automata theory as well as from a process algebra perspective.

[4] Bereichstheoretische Eigenschaften komplexer Spuren. (in German) Diplomarbeit / Master Thesis Nr. 1025, Institut für Informatik (IfI), Universität Stuttgart, August 1993 (36 pages).

Zusammenfassung

In dieser Arbeit werden die Ordnungseigenschaften des Monoids komplexer Spuren bezüglich der Präfixrelation untersucht. Die in \cite{gp92mfcs} eingeführte Präfixordnung, sowie die darin enthaltenen Ergebnisse bezüglich Vollständigkeit und Algebraizität werden vorgestellt. Der Hauptteil der Arbeit beschäftigt sich mit der Charakterisierung der Abhängigkeitsalphabete, für die das dazugehörige Monoid komplexer Spuren algebraisch ist (\cite{gp92open}) und liefert einen Beweis für die in \cite{die93mfcs} aufgestellte Vermutung dass *die Menge der komplexen Spuren über ein Abhängigkeitsalphabet genau dann algebraisch ist, wenn das Abhängigkeitsalphabet keine Kette von vier Knoten als induzierten Untergraphen enthält.* Der Nachweis dieser Äquivalenz erfolgt in mehreren Schritten und benutzt die Tatsache dass *ein Graph eine Kette von vier Knoten (ein Z-Graph) als induzierten Untergraphen enthält sobald der Graph und dessen Komplement zugleich zusammenhängend sind.*

Einleitung

Die vorliegende Diplomarbeit beschäftigt sich mit Ordnungseigenschaften partiell kommutativer Monoide, auch Spurenmonoide genannt. Diese mathematische Strukturen wurden von P. Cartier und D. Foata \cite{cf69} eingeführt, um gewisse kombinatorische Ordnungsprobleme zu lösen. Innerhalb der Informatik wurden sie als ein algebraisches Mittel zur natürlichen Modellierung der Semantik paralleler Prozesse als erstes durch die Pionierarbeit von A. Mazurkiewicz \cite{maz77} erkannt. Dabei entspricht die Multiplikation (Konkatenation) in diesen Monoiden der Hintereinanderausführung von Prozessen.

In der Zwischenzeit erfolgte eine systematische Untersuchung der Eigenschaften dieser Strukturen, die eine in sich zusammenhängende Theorie ergaben. Das Modell der endlichen Spuren wurde im Laufe der Zeit mehrmals erweitert, um verschiedene Aspekte paralleler Prozesse zu berücksichtigen. Die Notwendigkeit der Modellierung nicht-terminierender Prozesse hat zum Beispiel zur Erforschung unendlicher (tansfiniter) Spuren geführt. Um die bei der Hintereinanderausführung unendlicher Prozesse auftauchenden Probleme zu beseitigen, wurde daraufhin in \cite{die91} das Monoid komplexer Spuren als Faktormonoid bezüglich einer geeigneten Kongruenzrelation definiert.

In dieser Arbeit werden die Ordnungseigenschaften des Monoids komplexer Spuren bezüglich der Präfixrelation untersucht. Die in \cite{gp92mfcs} eingeführte Präfixordnung, sowie die darin enthaltenen Ergebnisse bezüglich Vollständigkeit und Algebraizität werden vorgestellt. Im einführenden Teil der Arbeit haben wir uns sehr nahe an die Darstellung der Spurentheorie aus den Monographien \cite{die90} und \cite{dr93} gehalten und die für uns relevanten Ergebnisse übernommen. Der Hauptteil der Arbeit beschäftigt sich mit der Charakterisierung der Abhängigkeitsalphabete, für die das dazugehörige Monoid komplexer Spuren algebraisch ist (\cite{gp92open}) und liefert einen Beweis für die in \cite{die93mfcs} aufgestellte Vermutung:

Die Menge der komplexen Spuren über ein Abhängigkeitsalphabet genau dann algebraisch ist, wenn das Abhängigkeitsalphabet keine Kette von vier Knoten als induzierten Untergraphen enthält.

Eine solche Kette nennen wir ein Z-Graph, in Anlehnung an die graphische Darstellung des Buchstabens Z. Der Nachweis dieser Äquivalenz erfolgt in mehreren Schritten und benutzt folgendes Ergebnis, das unabhängig von dem restlichen Teil der Arbeit interessant ist, nämlich:

Sind ein Graph und dessen Komplement zusammenhängend, so enthält der Graph eine Kette von vier Knoten (einen Z-Graphen) als induzierten Untergraphen.

Der von uns präsentierte Beweis dieser einfachen graphentheoretischen Asussage, der sowohl elementar als auch sehr anschaulich ist, konnte in der einschlägigen Literatur nicht ausfindig gemacht werden, weswegen wir uns darin bestätigt sehen die Originalität der Beweisführung zu beanspruchen.