Ideal Stream Algebra

Bernhard Möller

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany,
e-mail: moeller@uni-augsburg.de

Abstract. We provide some mathematical properties of *behaviours* of systems, where the individual elements of a behaviour are modeled by *ideals*, ie. downward closed directed subsets of a suitable partial order. It is well-known that the associated ideal completion provides a simple way of constructing algebraic cpos. An ideal can be viewed as a set of consistent finite or compact approximations of an object which itself may be infinite. A special case is the domain of streams where the finite approximations are the finite prefixes of a stream.
We introduce a special way of characterising behaviours through sets of relevant approximations. This is a generalisation of the technique we have used earlier for the case of streams. Given a set $P \subseteq M$ of a partial order $(M, \leq)$, we define

$$\mathsf{ide}\, P := \{Q^{\leq} : Q \subseteq P \text{ directed}\} ,$$

where $Q^{\leq} := \{x \in M : \exists\, y \in Q : x \leq y\}$ is the downward closure of Q. So $\mathsf{ide}\, P$ is the set of all ideals "spanned" by directed subsets of P. We prove a number of distributivity and monotonicity laws for ide and related operators. They are the basis for correct refinement of specifications into implementations. Various small examples illustrate that the operators lead to very concise while quite clear specifications.
Finally, we give a characterisation of safety and liveness and generalise the Alpern/Schneider decomposition lemma to arbitrary domains.
An extended example concerns the specification and transformational development of an asynchronous bounded queue.

Part I: Introduction

1 Origin and Goals

The context of this work is deductive program design, in which implementations are derived from specifications by semantics-preserving deduction rules. Examples of this paradigm are transformational program development (eg. [53, 8]) and

This research was partially sponsored by Esprit Working Group 8533 NADA — New Hardware Design Methods

the refinement calculus (eg. [16, 23, 5, 2, 46, 47]). There is a growing conviction that this paradigm is most efficient when based on algebraic rather than purely logical frameworks. The aim there is to make program specification and calculation more concise and perspicuous by compacting logic into algebra as much as possible. For sequential programs this is demonstrated eg. in [39, 41, 8].

In the parallel case, to some extent the work reported in [50, 14] can be viewed as falling into the algebraic realm; purely algebraic approaches are presented in [31, 56]. The present paper presents a particular approach to streams (see eg. [30, 49, 12] and [62] for a recent survey). It centres around the order-theoretic view of streams and other semantic objects as used in denotational semantics. In addition to order theory we use a suitable algebra of formal languages [39] in reasoning about streams.

To exhibit a certain uniformity we use the same calculational style of reasoning in the parts treating the mathematical background as in the program derivations proper.

2 Streams and Ideals

The basic tool in our approach is the prefix order on finite words in A^* over some alphabet A of basic actions, data or states. These words are considered as initial parts of system traces. A trace language is directed w.r.t. this order iff it is totally ordered by it. Therefore ideals, ie. prefix closed directed sets of traces, are a suitable representation of finite and infinite streams.

It is well known that the space of streams under the prefix ordering is isomorphic to the ideal completion of the set of finite streams. Since, however, ideals are just particular trace languages, we can use all operations on formal languages for their manipulation. A large extent of this is covered by conventional regular algebra. Moreover, we can apply the tools developed for quite different purposes in a number of papers on algebraic calculation of graph, pointer and sorting algorithms (see [39, 41] and the references there). Finally, we do not need additional mechanisms for dealing with fairness; rather, fairness is made explicit within the generating expressions for trace languages.

Using regular expressions rather than automata or transition systems gives considerable gain in conciseness and clarity, both in specification and calculation. While this has long been known in the field of syntax analysis, most approaches to the specification of concurrency stay with the fairly detailed level of automata, thus leading to cumbersome and imperspicous expressions. Other approaches use logical formulas for describing sets of traces; these, too, can become very involved. By extracting a few important concepts and coming up with closed expressions for them one can express things in a more structured and concise form. This is done here using regular and regular-like expressions with their strong algebraic properties. The approach can also be nicely tied in with temporal and modal operators (see [45]).

Another advantage of our approach is that we can do with simple set-theoretic notions thus avoiding most of the overhead of domain theory. By this, the approach also is completely orthogonal w.r.t. nesting of data structures, ie. it admits streams of functions, streams of sets, sets of streams, streams of streams etc. without problems.

3 A Simple Soda Machine

To show the style of our approach and in order to better motivate the technicalities to come we first give a number of examples with informal explanations. The precise definitions will be given in later sections.

We start with the description of a simple soda machine. It accepts half dollars and quarters and emits a can of soda after having received a half dollar's worth in coins. Let h and q denote the events of receiving a half dollar and a quarter, respectively, and c the event of emitting a can of soda. Then the behaviour of that machine is described by the regular-like expression

$$((h \cup q \bullet q) \bullet c)^\omega \ ,$$

where $\bullet$ is concatenation and $\_^\omega$ denotes infinite repetition. Each expression of this kind denotes a set of (finite or infinite) streams; in the case of the soda machine all these streams are infinite.

In the above expression, the iterated subexpression $(h \cup q \bullet q) \bullet c$ states the following safety properties: the customer must insert the correct amount of money and is not allowed to insert further money before delivery of the can. The infinite repetition $\_^\omega$ combines safety and liveness aspects: it expresses the correct order of insert/deliver cycles, a safety property, and expresses the temporal aspect of eventuality (see eg. [22]): it guarantees that after insertion of a sufficient amount of money eventually a can is delivered and the machine is ready to accept further orders.

We prefer to leave states implicit as long as possible, since frequently regular expressions are clearer and more concise than the corresponding descriptions by accepting automata (Büchi automata in the case of infinite repetition, see eg. [52, 65, 66]).

4 Fairness

Other eventuality properties can already be expressed by Kleene's finite repetition operators $\_^*$ and $\_^+$. To exemplify this, we describe a scheduler for unboundedly fair merging of input from two channels. It is modeled as an infinite stream over the alphabet $\{0,1\}$, where 0 denotes choice from the left and 1 choice from the right input channel of the merge module. A sequence in which there is at least once a choice from the left followed eventually by a choice from the right is described by the regular expression $0^+ \bullet 1$. By adding the symmetric requirement and, again, infinite repetition to drive the single cycles, we get the following description of the set of streams that model the behaviour of a fair scheduler:

$$\mathcal{SCHED} \overset{\text{def}}{=} (0^+ \bullet 1 \cup 1^+ \bullet 0)^\omega \ .$$

The "local eventuality" is here expressed by the finiteness of $\_^+$, whereas the infinite repetition $\_^\omega$ again adds liveness and "global eventuality".

Arbitrary (and hence possibly non-fair) merge would be obtained by replacing this scheduler by $(0 \cup 1)^\omega$.

The reason why fairness does not cause problems in our approach is that fairness constraints are expressed using the star operation which has a simple recursive definition using least fixpoints w.r.t. the inclusion ordering on sets of streams, whereas there are continuity problems w.r.t. extensions of the prefix order to sets of streams. This is due to the fact that the prefix order has operational traits and unbounded fairness is operationally not feasible, whereas the inclusion ordering is purely descriptive and hence does not face this problem. It is adequate for proving properties of sets of streams; when it comes to implementation, of course operationally feasible descendants have to be used.

We prefer to state fairness assumptions explicitly, since this gives much greater flexibility than building them into the underlying semantic framework (such as eg. in [17]).

5 Channels

Another aspect of fairness and eventuality is exhibited in the description of channels as used in many protocol specifications. The channels are faulty, but fair in the sense that after an unbounded but finite number of faulty transmissions they will at least once transmit correctly.

We will describe their behaviour using streams of functions. Each such function models the transmission behaviour at one particular instance of time. The identity function *id* models correct transmission, *fail* transforms any message into an "error element" and *skip* transforms any message into empty output. Let in the sequel stand the sub/superscript i for $*$ (unbounded but finite repetition) or $\leq k$ for some $k \in \mathbb{N}$ (bounded repetition). Then the following specifications express unbounded and bounded fairness, respectively.

A possibly corrupting but fair channel is described by

$$\mathit{cchan}_i \stackrel{\text{def}}{=} (\mathit{fail}^i \bullet \mathit{id})^\omega \ ,$$

a possibly lossy but fair channel by

$$\mathit{lchan}_i \stackrel{\text{def}}{=} (\mathit{skip}^i \bullet \mathit{id})^\omega$$

and a possibly lossy and corrupting but fair channel by

$$\mathit{lcchan}_i \stackrel{\text{def}}{=} ((\mathit{skip} \cup \mathit{fail})^i \bullet \mathit{id})^\omega \ .$$

An unfair corrupting channel is

$$\mathit{arbchan} \stackrel{\text{def}}{=} (\mathit{fail} \cup \mathit{id})^\omega \ .$$

This kind of channel descriptions has been used in [42] for a very concise algebraic correctness proof of the alternating bit protocol.

6 Two Stream-Based Models of Systems

6.1 Modules as Stream-Processing Functions

A stream-processing function (SPF) is a function from tuples of input streams to tuples of output streams (see eg. [30, 12]). In the case of synchronous systems this may equivalently be replaced by a function from a stream of input tuples to a stream of output tuples.

In the SPF view each module is described as an SPF. The advantage of this model is that it allows easy definitions of various composition operations for modules and hence lends itself to a modular structuring of large systems.

The disadvantage in the description of asynchronous systems is that the separation between input and output streams loses causal information, viz. which input triggered which output. This gives rise to the (in)famous merge anomaly [9] which can be fixed by re-introducing time information into the streams. It has to be expressed which elements of a stream are considered to belong to the same time interval. This can be done using explicit time ticks [11, 63] or streams of sequences where each sequence lists the elements that belong to one time interval [32].

6.2 Trace Models

In the trace view, the overall system is described by admissible sequentialisations of actions during system runs (interleaving semantics). If the structure of non-deterministic branching is preserved, one obtains tree-like semantic domains such as used in CCS [36], CSP [26] and process algebra [3]. In the simplest case, however, the trace structure is a *set* of streams (see also [29]) which we term a *behaviour*. In this view, a stream in A^∞ is a complete record of one possible system run with all system actions interleaved.

Eg. for a CSP-like view one uses the alphabet $A = C \times V$ of basic actions, where C is a set of channel names and V a set of values that are transmitted along the channels. Then the streams in A^∞ are complete records of system runs with all channel activities interleaved.

The advantage of this view is that it keeps track of the causality between input and output; hence the merge anomaly does not arise.

The disadvantage is a loss in modularity, since only the overall system is described directly. Modularisation can be re-introduced, though, by restricting attention to subsets of channels.

Part II: The Algebra of Ideals

This part reviews a few order-theoretic notions and provides some auxiliary facts. It then goes on to develop the ideal-theoretic basis of our approach. On first reading, this part may be skipped and consulted later for details as the need arises.

7 Mathematical Background

7.1 Order-Theoretic Preliminaries

In this section we repeat some basic notions from the theory of partial orders. Some useful properties of the operations introduced are given in the Appendix. Proofs not contained in the present paper can be found in [42].

For partially ordered set $(M, \leq)$ and $N \subseteq M$ we define the *proper* and *improper downward closure* by

$$N^{<} \overset{\text{def}}{=} \{y \in M : \exists\, x \in N : y < x\}$$
$$N^{\leq} \overset{\text{def}}{=} \{y \in M : \exists\, x \in N : y \leq x\} \;=\; N \cup N^{<}$$

where $y < x \Leftrightarrow y \leq x \wedge x \neq y$.

The set of *maximal* elements of $N \subseteq M$ is defined by

$$\mathsf{max}\, N \overset{\text{def}}{=} N \backslash N^{<} \;.$$

We now extend the order $\leq$ to a relation on subsets of M by

$$N \leq P \overset{\text{def}}{\Leftrightarrow} N \subseteq P^{\leq} \;.$$

This is the angelic half of the Egli-Milner pre-order [54]. In particular, $N^{\leq} \leq N$.

Since $\leq$ generally is only a pre-order between sets, we are interested in the induced equivalence relation

$$N \sim P \overset{\text{def}}{\Leftrightarrow} N \leq P \wedge P \leq N \;.$$

A subset $N \subseteq M$ is a *cone* if it is downward closed, ie. if $N^{\leq} \subseteq N$. Hence on cones $\leq$ and $\subseteq$ coincide; in particular, $\leq$ is a partial order on cones.

Since M is a cone and the intersection of cones is a cone again, the set of all cones forms a complete lattice under inclusion. It is isomorphic to the angelic or Hoare power domain [58] over $(M, \leq)$. However, we are not going to use that domain.

7.2 Pointwise Extension

In the sequel we will define many functions on single points of M and lift them to subsets of M by *pointwise extension*, ie. by setting, for $f : M \to M$ and $N \subseteq M$,

$$f(N) \overset{\text{def}}{=} \{f(x) : x \in N\} \;.$$

These pointwise extended functions distribute through arbitrary unions and hence are monotonic w.r.t. inclusion and strict w.r.t. $\emptyset$. We will also use this mechanism to lift these functions a further level to sets of subsets of M.

Pointwise extensions inherit *linear laws*. These are laws of the following form:

– Equational laws in which all variables occur exactly once on both sides of the equality sign. Examples are the laws of neutrality, associativity and commutativity over a groupoid.
– Implications with element relations as atoms in which all variables occur exactly once on both sides of the implication sign. In the inherited form the variables for elements turn into variables for non-empty sets and the element relations turn into inclusions. An example is

$$s \bullet t \in \varepsilon \Rightarrow s \in \varepsilon \wedge t \in \varepsilon$$

which lifts to

$$S \neq \emptyset \wedge T \neq \emptyset \wedge S \bullet T \subseteq \varepsilon \;\Rightarrow\; S \subseteq \varepsilon \wedge T \subseteq \varepsilon\ .$$

7.3 Directed Sets and the Ideal Completion

A subset $N \subseteq M$ is *directed* if every finite subset of N has an upper bound in N. Equivalently, N is directed if $N \neq \emptyset$ and any two elements of N have a common upper bound in N. Hence every two elements of a directed set are consistent in that they approximate a common element.

For $P \subseteq M$ we denote by $\mathsf{dir}\, P$ the set of all directed subsets of P. Note that the operation dir is monotonic w.r.t. inclusion. Some further properties of dir can be found in the Appendix.

To tie our approach in with domain-theoretic notions (see eg. [64] we recall the ideal completion (cf. eg. [6, 19]). Consider an ordered set $(M, \leq)$. An *ideal* is a directed cone. The set of all ideals is denoted by $I(M)$.

The partial order $(M, \leq)$ is called *complete* or a *cpo* iff every directed set $D \subseteq M$ has a supremum (or least upper bound) $\sqcup\, D \in M$. An element x of M is *finite* (*compact*) iff for every directed set $D \subseteq M$ with $x \leq \sqcup D$ we have also $x \leq z$ for some $z \in D$. Equivalently, x is finite iff for every ideal $I \subseteq M$ with $x \leq \sqcup I$ we have $x \in I$. $(M, \leq)$ is *algebraic* iff every element of M is the supremum of a directed set of finite elements. A non-finite element of an algebraic set is called a *limit point* or an *infinite element*. With these notions one has

Theorem 7.1 *1. The set $(I(M), \subseteq)$ ordered by set inclusion is a cpo and algebraic, the finite elements being the* principal ideals *$x^{\leq}$ for $x \in M$. The mapping $\iota : x \mapsto x^{\leq}$ is an embedding of M into $I(M)$.*
2. For every monotonic mapping $h : M \to P$ into a cpo $(P, \leq)$ there is a unique continuous mapping $\overline{h} : I(M) \to P$ extending h, ie. with $\overline{h}(x^{\leq}) = h(x)$. $\overline{h}$ is given by $\overline{h}(I) = \sqcup\, h(I)$ for $I \in I(M)$; hence $\overline{h}(D^{\leq}) = \sqcup\, h(D)$ for directed $D \subseteq M$.

The ordered set $(I(M), \subseteq)$ is called the *ideal completion* of $(M, \leq)$. We set $M^{\infty} \overset{\text{def}}{=} I(M)$. An ideal in M^{∞} is non-compact iff it does not have a maximal (and hence greatest) element.

8 Streams as Ideals

We now make our notion of streams precise. Assume an alphabet A of atomic actions, data or states. Then, as usual, A^* is the set of all finite words over A. By ε we denote the empty word, whereas concatenation is denoted by $\bullet$. A subset of A^* is called a *(formal) language*.

A word u is a *prefix* of a word v, written $u \sqsubseteq v$, iff there is a word w such that $u{\bullet}w = v$. It is well-known that this defines a partial order on words which is even well-founded. Moreover, ε is the least element in this order. The corresponding strict-order is denoted by $\sqsubset$. A cone of $(A^*, \sqsubseteq)$ is then a prefix-closed language. Note that every non-empty cone contains ε.

A few properties we shall use are the following (where $x, y, u, v, w \in A^*$ and $U, V \subseteq A^*$):

$$v \sqsubseteq w \Leftrightarrow u \bullet v \sqsubseteq u \bullet w \;, \tag{1}$$

$$u \sqsubseteq w \wedge v \sqsubseteq w \Rightarrow u \sqsubseteq v \vee v \sqsubseteq u \;, \tag{2}$$

$$V \neq \emptyset \;\Rightarrow\; (U \bullet V)^{\sqsubseteq} = U^{\sqsubseteq} \cup U \bullet V^{\sqsubseteq} \;. \tag{3}$$

Property (2) is also called *local linearity*.

Informally, a stream over A is a finite or infinite sequence of elements of A. The basis of our approach is the observation that such a stream is completely characterised by the set of its finite prefixes. This set is downward closed w.r.t. $\sqsubseteq$, ie. a cone. Moreover, it is directed, since in the partial order $(A^*, \sqsubseteq)$ by local linearity the directed sets can be characterised another way:

Lemma 8.1 *$D \subseteq A^*$ is directed w.r.t. $\sqsubseteq$ iff D is totally ordered by $\sqsubseteq$, ie. iff for any two elements $u, v \in D$ we have $u \sqsubseteq v$ or $v \sqsubseteq u$.*

Hence an ideal of $(A^*, \sqsubseteq)$ is a totally and prefix-closed non-empty language. Note that every ideal contains ε. Therefore an ideal is a set of words of increasing length "growing at the right end". This set may be finite or infinite. A simple example is, for $a \in A$, the infinite ideal

$$a^* = \{\varepsilon,\; a,\; a \bullet a,\; a \bullet a \bullet a,\; a \bullet a \bullet a \bullet a,\; \ldots\} \;.$$

We identify a stream with the set of its finite prefixes. By the above, this set is an ideal of $(A^*, \sqsubseteq)$. Therefore we call the elements of A^∞ *streams* over A. It should be noted that the compact elements of A^∞ correspond to the elements of A^*; hence, for countable A, the set $(A^\infty, \subseteq)$ has a countable basis of finite elements and therefore is countably algebraic. The *length* of stream S is denoted by $|S|$; it coincides with its cardinality minus one. Let us give a characterisation of infinite streams:

Lemma 8.2 *A stream S is infinite iff $\mathsf{max}\, S = \emptyset$.*

Proof. First, by linearity of the prefix order on a stream and by its well-foundedness, an infinite stream cannot have a maximal element. The reverse implication is provided by Lemma 12.1.2. $\square$

The compact elements of A^∞ correspond to the elements of A^*, whereas the non-compact elements are precisely the (cardinally) infinite ideals. They correspond to infinite sequences over A.

To resume our previous example, the ideal

$$a^* = \{\varepsilon,\ a,\ a \bullet a,\ a \bullet a \bullet a,\ a \bullet a \bullet a \bullet a,\ \ldots\}$$

is the limit (supremum) of the set of finite ideals

$$\{\{a^i : i \leq n\} : n \in \mathbb{N}\}$$

corresponding to the $\sqsubseteq$-increasing set

$$\{a^n : n \in \mathbb{N}\}$$

of finite words. It may thus be viewed as a representation of the infinite stream of as. This observation is the main motivation for our approach; it allows us to work with infinite streams by manipulating their sets of finite approximations, since in the ideal completion each (finite or infinite) element is *identified* with the set of its finite approximations. This allows carrying over all laws from the algebra of formal languages to streams. Of course, the fact that the set of finite and infinite streams is isomorphic to the ideal completion of the set of finite streams is well-known; what is new here is the direct algebraic manipulation of the ideals using those laws.

While our approach was motivated by the particular case of streams, we will perform the mathematical development as far as possible for general ideal completions.

9 A Setting for Non-Interleaving Semantics

To stress that latter point and to illustrate our approach with a different setting we now sketch how partial-order semantics, allowing true concurrency, can be accommodated in our setting.

Let E be a set of *events*. Then a *history* over E is a partial order $(F, \preceq)$ with a finite subset $F \subseteq E$ of events. The order $\preceq$ models temporal/causal dependence. Two events not related by $\preceq$ are considered as parallel/concurrent.

Let now $H(E)$ be the set of all histories over E. We define an approximation ordering $\leq$ on $H(E)$ by

$$\begin{aligned}(F_1, \preceq_1) \leq (F_2, \preceq_2) \overset{\text{def}}{\Leftrightarrow}\ & F_1 \subseteq F_2 \wedge \\ & \preceq_1\ =\ \preceq_2 \cap F_1 \times F_2\ \wedge \\ & \forall\, x \in F_1 : x^{\preceq_1} = x^{\preceq_2}\ \wedge \\ & \forall\, y \in F_2 : \exists\, x \in F_1 : x \preceq_2 y\ .\end{aligned}$$

This is the appropriate generalisation of the prefix relation on words to histories. It means that F_1 is embedded as a cone into F_2 and F_2 may only add “later” events. It is straightforward to check that this indeed defines a partial order. The least element is $(\emptyset, \emptyset)$.

A *chronicle* now is an ideal in $(H(E), \leq)$, and infinite chronicles generalise infinite streams. The case of streams is retrieved if one only considers histories that are linearly ordered by $\preceq$; in that case $\preceq$ corresponds directly to $\sqsubseteq$. In the present paper, we shall not pursue this example further, though.

10 Behaviours and Refinement

Our application of ideals will be the description of systems. To model non-determinacy, we define a *behaviour* to be a set of ideals.

It should be noted that using *sets* of ideals as behaviours allows only "trace-like" semantics in which there is no distinction between internal and external non-determinacy. The algebraic reflection of this is that concatenation, our sequencing operation, distributes through union both from the left and from the right. In algebraic approaches to CCS-like systems (see eg. [36, 3]) only one of these distributivities holds. This results in models with tree-like objects that reflect the non-deterministic branching structure in time. This detailed record is lost by admitting both distributivities rather than just one.

The set of finite prefixes of a behaviour $\mathcal{B}$ is

$$\mathsf{pref}\,\mathcal{B} \overset{\text{def}}{=} \bigcup \mathcal{B} \ .$$

Clearly, pref distributes through union and hence is $\subseteq$-monotonic.

As our refinement relation we choose inclusion, ie. behaviour $\mathcal{B}$ *refines* behaviour $\mathcal{C}$ if $\mathcal{B} \subseteq \mathcal{C}$. For instance, given a property $P \subseteq M$, the set $\mathsf{ide}\,P$ of ideals satisfying P, is a behaviour. To allow correct local refinements one therefore has to ensure monotonicity of all operations w.r.t. inclusion.

Example 10.1 We resume the example from Section 4 and show that bounded fairness refines unbounded fairness: since all operators involved are monotonic w.r.t. inclusion, we obtain from $a \bullet a^{\leq k} \subseteq a^+$ for $a \in A$ that

$$(0 \bullet 0^{\leq k} \bullet 1 \cup 1 \bullet 1^{\leq k} \bullet 0)^\omega \subseteq \mathcal{SCHED} \ .$$

$\square$

11 Describing Behaviours by Properties

We want to characterise ideals by certain sets of "relevant" or "admissible" finite approximations. Such a set, ie. a subset of our overall partially ordered set M, is called a *property* in this connection.

In the particular case of streams the finite approximations are "snapshots" in the form of finite words in A^*. Assume a set $U \subseteq A^*$ of admissible snapshots. If a stream contains a subset $D \subseteq U$ of snapshots then D has to be directed. However, there may be arbitrary "gaps" between the snapshots in D. To reconstruct the stream we therefore have to "fill in the details" between the snapshots. This is done by taking the prefix closure $D^{\sqsubseteq}$. Hence we define the set of streams, ie. the behaviour, spanned by snapshot set U as

$$\mathsf{str}\, U \overset{\text{def}}{=} \{D^{\sqsubseteq} : D \in \mathsf{dir}\, U\} \ .$$

This is the set of streams that "interpolate" consistent snapshots in U. A related notion occurs in [20]; the connection will be made precise in Section 12.

We generalise this to arbitrary partial orders and their ideal completions. Let $(M, \leq)$ be the partial order of finite approximations. For property $P \subseteq M$ we now define by

$$\mathsf{ide}\, P \overset{\text{def}}{=} \{D^{\leq} : D \in \mathsf{dir}\, P\}$$

the set of all ideals "spanned" by directed subsets of P. Note that $\mathsf{ide}\, M = I(M)$. Moreover, ide is monotonic w.r.t. inclusion. A different characterisation of ide is given by

Lemma 11.1 *For $I \in I(M)$ and $Q \subseteq M$ the following statements are equivalent:*

1. $I \in \mathsf{ide}\, Q$.
2. $I \subseteq (I \cap Q)^{\leq}$.
3. $I = (I \cap Q)^{\leq}$.

Proof. The equivalence of 2 and 3 is obvious by monotonicity of $^{\leq}$ and downward closedness of I.
(1 ⇒ 2) Suppose $I = D^{\leq}$ for $D \in \mathsf{dir}\, Q$.

$$\begin{array}{ll} & I \\ = & \quad \{\!\![\text{ assumption }]\!\!\} \\ & D^{\leq} \\ = & \quad \{\!\![\text{ since } D \subseteq Q \ ]\!\!\} \\ & (D \cap Q)^{\leq} \\ \subseteq & \quad \{\!\![\text{ monotonicity }]\!\!\} \\ & (D^{\leq} \cap Q)^{\leq} \\ = & \quad \{\!\![\text{ assumption }]\!\!\} \\ & (I \cap Q)^{\leq} \ . \end{array}$$

(3 ⇒ 1) Since I is directed, so is $(I \cap Q)^{\leq}$. By Lemma 30.5.3 also $I \cap Q$ is directed and the claim follows. □

We have the following distributivity property for ide:

Lemma 11.2 *Consider $N, P \subseteq M$. Then*

$$\mathsf{ide}\,(N \cup P) = \mathsf{ide}\, N \cup \mathsf{ide}\, P \ .$$

Proof. $I \in \mathsf{ide}\,(N \cup P)$

$\Leftrightarrow$ $\{\![$ by Lemma 11.1 $]\!\}$

$I = (I \cap (N \cup P))^{\leq}$

$\Leftrightarrow$ $\{\![$ distributivity of $\cap$ over $\cup$ and Lemma 30.1.1 $]\!\}$

$I = (I \cap N)^{\leq} \cup (I \cap P)^{\leq}$

$\Rightarrow$ $\{\![$ by directedness of I, Lemma 30.5.2 and Lemma 30.3.3 $]\!\}$

$I = (I \cap N)^{\leq} \vee I = (I \cap P)^{\leq}$

$\Leftrightarrow$ $\{\![$ by Lemma 11.1 $]\!\}$

$I \in \mathsf{ide}\,N \vee I \in \mathsf{ide}\,P$.

The reverse inclusion follows by monotonicity of ide.
Another proof can be given using Lemma 30.5.5. □

This also shows once again the monotonicity of ide. We have even

Corollary 11.3 $N \subseteq P \Leftrightarrow \mathsf{ide}\,N \subseteq \mathsf{ide}\,P$.

Proof. The inclusion from right to left is part of Theorem 7.1.1 via the principal ideals $x^{\leq}$ for $x \in M$. □

It should be noted, however, that ide only distributes through finite unions and hence is not "continuous". For an instance of this see Example 13.3 below.

Lemma 11.4 *We have the following properties concerning downward closure:*

1. $I \in \mathsf{ide}\,(P^{\leq}) \Leftrightarrow I \subseteq P^{\leq}$.
2. $\mathsf{pref}\;\mathsf{ide}\,P = P^{\leq}$.
3. $\mathsf{ide}\,Q \subseteq \mathsf{ide}\,Q^{\leq}$. *The reverse inclusion is not valid.*

Proof. 1. ($\Rightarrow$) Assume $I = D^{\leq}$ for $D \in \mathsf{dir}\,(P^{\leq})$. Then, by monotonicity and idempotence of $^{\leq}$ we get $D^{\leq} \subseteq (P^{\leq})^{\leq} = P^{\leq}$, ie. $I \subseteq P^{\leq}$.
($\Leftarrow$) Straightforward, since $I \subseteq P^{\leq}$ implies $I \in \mathsf{dir}\,(P^{\leq})$ and $I = I^{\leq}$.
2. The inclusion $\subseteq$ is straightforward. For the reverse consider $y \in P^{\leq}$. There is $x \in P$ with $y \leq x$. But then $y \in x^{\leq} \in \mathsf{ide}\,P$.
3. Immediate from $Q \subseteq Q^{\leq}$ and monotonicity of ide. For a counterexample to the reverse inclusion see Example 13.1.

□

12 Maximal and Infinite Ideals

12.1 Maximal Ideals

Frequently one is interested in processes that continue as long as possible. These are modeled by ideals which are maximal w.r.t. $\leq$ or, equivalently, w.r.t. inclusion. We therefore give a characterisation of maximal ideals. For a behaviour $\mathcal{B}$ we denote the subset of maximal ideals by $\mathsf{max}\,\mathcal{B}$; this agrees with the definition in Section 7.1, and hence all our laws in the Appendix apply.

Lemma 12.1 *Suppose $I \in I(M)$ and $N \subseteq M$. Then*

1. $x \in \mathsf{max}\, I \Leftrightarrow I = x^{\leq}$.
2. $\mathsf{max}\, I = \emptyset \Rightarrow I$ *infinite.*
3. $\mathsf{max}\, N = \emptyset \wedge I \in \mathsf{max}\,\mathsf{ide}\, N \Rightarrow \mathsf{max}\, I = \emptyset$.

Proof. 1. ($\Rightarrow$) We only need to show $I \subseteq x^{\leq}$; the other inclusion follows from downward closure of I. Suppose $y \in I$. By directedness of I there is $z \in I$ with $x \leq z$ and $y \leq z$. Maximality of x implies $z = x$ and hence $y \leq x$.
($\Leftarrow$)

$$\begin{array}{ll} & \mathsf{max}\, I \\ = & \quad \{\!\![\text{ by assumption }]\!\!\} \\ & \mathsf{max}\, x^{\leq} \\ = & \quad \{\!\![\text{ by Lemma 30.2.1 }]\!\!\} \\ & (x^{\leq})^{\leq} \backslash (x^{\leq})^{<} \\ = & \quad \{\!\![\text{ by Lemma 30.1.2 }]\!\!\} \\ & x^{\leq} \backslash x^{<} \\ = & \quad \{\!\![\text{ by Lemma 30.2.1 }]\!\!\} \\ & \mathsf{max}\, x \\ = & \quad \{\!\![\text{ irreflexivity of } <]\!\!\} \\ & \{x\} \ . \end{array}$$

2. Every non-empty finite set has a maximal element.
3. Suppose $\mathsf{max}\, I \neq \emptyset$, say $x \in \mathsf{max}\, I$. By 1 then $I = x^{\leq}$ and by $I \in \mathsf{ide}\, N$ we get $x \in N$. Since $\mathsf{max}\, N = \emptyset$, there is $y \in N$ with $x \leq y$ and $y \neq x$. But then $y^{\leq} \in \mathsf{ide}\, N$ and hence, by Theorem 7.1.1, we have $x^{\leq} \subseteq y^{\leq} \wedge x^{\leq} \neq y^{\leq}$. This is a contradiction to $I \in \mathsf{max}\,\mathsf{ide}\, N$.

□

12.2 Infinite Ideals

Motivated by Lemma 12.1.2 we define, for a behaviour $\mathcal{B}$, the set of its infinite ideals as

$$\mathsf{inf}\, \mathcal{B} \overset{\text{def}}{=} \{I \in \mathcal{B} : \mathsf{max}\, I = \emptyset\} \ .$$

For general domains, this is a bit of a misnomer, since there may well be infinite ideals *with* maximal elements. However, we will single out a particular class of domains where this cannot occur and work mostly with these, so that the terminology will be justified. Clearly, inf distributes through arbitrary union and intersection:

$$\mathsf{inf}\,(\bigcup_{i \in I} \mathcal{B}_i) = \bigcup_{i \in I} \mathsf{inf}\, \mathcal{B}_i \ , \tag{4}$$

$$\mathsf{inf}\,(\bigcap_{i \in I} \mathcal{B}_i) = \bigcap_{i \in I} \mathsf{inf}\, \mathcal{B}_i \ . \tag{5}$$

Now Lemma 12.1.3 can be restated as

$$\mathsf{max}\, N = \emptyset \Rightarrow \mathsf{max}\,\mathsf{ide}\, N \subseteq \mathsf{inf}\,\mathsf{ide}\, N \ .$$

The reverse inclusion is generally not valid. For a counterexample choose $M = \mathbb{N} \cup \{\infty\}$ with the usual ordering and consider the ideal $\mathbb{N} \in I(M)$. We have $\mathsf{max}\,\mathbb{N} = \emptyset$, but $\mathbb{N} \notin \mathsf{max}\, I(M)$, since $\mathbb{N} \subseteq M \in I(M)$ and $\mathbb{N} \neq M$.

We call a partial order $(M, \leq)$ max-*determined* if

$$\mathsf{inf}\, I(M) \subseteq \mathsf{max}\, I(M) \ .$$

12.3 Refinement Laws

Now we clarify the relation between inf ide and max ide and investigate monotonicity and distributivity of the max ide, inf ide and max inf operations, which is important for refinement. First we note

Lemma 12.2 *For $N, P \subseteq M$,*

$$\mathsf{inf}\,\mathsf{ide}\, N \cup P = \mathsf{inf}\,\mathsf{ide}\, N \cup \mathsf{inf}\,\mathsf{ide}\, P \ .$$

In particular, inf ide *is monotonic w.r.t. inclusion.*

Proof. Immediate from Lemma 11.2 and equation (4). □

Concerning maximal ideals we have

Lemma 12.3 *Let $(M, \leq)$ be* max-*determined. Then, for $N, P \subseteq M$,*

1. $\mathsf{inf}\,\mathsf{ide}\, N \subseteq \mathsf{ide}\, N \cap \mathsf{max}\, I(M) \subseteq \mathsf{max}\,\mathsf{ide}\, N$.
2. $\mathsf{max}\,\mathsf{ide}\, N = \mathsf{inf}\,\mathsf{ide}\, N \cup \mathsf{ide}\,\mathsf{max}\, N$.
3. $\mathsf{max}\, N = \emptyset \Rightarrow \mathsf{inf}\,\mathsf{ide}\, N = \mathsf{ide}\, N \cap \mathsf{max}\, I(M) = \mathsf{max}\,\mathsf{ide}\, N$.
4. $\mathsf{inf}\,\mathsf{ide}\, N \cup P = \mathsf{inf}\,\mathsf{ide}\, N \cup \mathsf{inf}\,\mathsf{ide}\, P$.
   In particular, inf ide *is monotonic w.r.t. inclusion.*
5. $N = \mathsf{saf}\, N \Rightarrow \mathsf{inf}\,\mathsf{ide}\,(N \cap P) = \mathsf{inf}\,\mathsf{ide}\, N \cap \mathsf{inf}\,\mathsf{ide}\, P$.
6. $\mathsf{max}\, N = \emptyset \wedge N \subseteq P \Rightarrow \mathsf{max}\,\mathsf{ide}\, N \subseteq \mathsf{max}\,\mathsf{ide}\, P$.
7. $\mathsf{max}\, N = \mathsf{max}\, P = \emptyset \Rightarrow \mathsf{max}\,\mathsf{ide}\,(N \cup P) = \mathsf{max}\,\mathsf{ide}\, N \cup \mathsf{max}\,\mathsf{ide}\, P$.
8. *If N and P are cones with $\mathsf{max}\, N = \mathsf{max}\, P = \mathsf{max}\,(N \cap P) = \emptyset$ then* $\mathsf{max}\,\mathsf{ide}\,(N \cap P) = \mathsf{max}\,\mathsf{ide}\, N \cap \mathsf{max}\,\mathsf{ide}\, P$.

Proof. 1. $I \in \mathsf{inf}\,\mathsf{ide}\, N$

$\Leftrightarrow$ {[definition]}

$I \in \mathsf{ide}\, N \wedge \mathsf{max}\, I = \emptyset$

$\Rightarrow$ {[since $(M, \leq)$ is max-determined]}

$I \in \mathsf{ide}\, N \wedge I \in \mathsf{max}\, I(M)$

$\Rightarrow$ {[by Lemma 30.2.3, since $\mathsf{ide}\, N \subseteq I(M)$]}

$I \in \mathsf{max}\,\mathsf{ide}\, N$.

2. ($\subseteq$) Suppose $I \in \mathsf{max}\,\mathsf{ide}\,N$. If $\mathsf{max}\,I = \emptyset$, then $I \in \mathsf{inf}\,\mathsf{ide}\,N$ by definition. Otherwise $\mathsf{max}\,I$ is a singleton, say $\mathsf{max}\,I = \{x\}$, and $I = x^{\leq}$. It follows that $x \in N$. For $y \in N$ with $x \leq y$ we have $x^{\leq} \subseteq y^{\leq} \in \mathsf{ide}\,N$, so that $x^{\leq} = y^{\leq}$ by maximality of $I = x^{\leq}$. Hence also $x = y$. This shows $x \in \mathsf{max}\,N$, so that $I = x^{\leq} \in \mathsf{ide}\,\mathsf{max}\,N$.
   ($\supseteq$) $\mathsf{inf}\,\mathsf{ide}\,N \subseteq \mathsf{max}\,\mathsf{ide}\,N$ was shown in 1. Suppose now $I \in \mathsf{ide}\,\mathsf{max}\,N$, say $I = x^{\leq}$ with $x \in \mathsf{max}\,N$, and $I \subseteq J \in \mathsf{ide}\,N$, say $J = D^{\leq}$ for $D \in \mathsf{dir}\,N$. Consider $y \in J$. By directedness of J there is a $z \in J$ with $x, y \leq z$. By $J = D^{\leq}$ there is a $u \in D$ with $z \leq u$. Hence also $x, y \leq u$. By $D \subseteq N$ and $x \in \mathsf{max}\,N$ we get $x = u$. So $y \leq x$ and hence $y \in x^{\leq} = I$. Altogether, $J \subseteq I$ and hence $J = I$. So $I \in \mathsf{max}\,\mathsf{ide}\,N$.
3. Assume $\mathsf{max}\,N = \emptyset$. Then by Lemma 12.1.3 $\mathsf{max}\,\mathsf{ide}\,N \subseteq \mathsf{inf}\,\mathsf{ide}\,N$ and the equalities follow from 1.
4. $\mathsf{max}\,\mathsf{ide}\,N$

   $=$ {[by 3]}

   $\mathsf{ide}\,N \cap \mathsf{max}\,I(M)$

   $\subseteq$ {[by assumption $N \subseteq P$ and monotonicity of ide]}

   $\mathsf{ide}\,P \cap \mathsf{max}\,I(M)$

   $\subseteq$ {[by 3]}

   $\mathsf{max}\,\mathsf{ide}\,P$.
5. We aim at an application of Lemma 30.2.4. Assume $I \in \mathsf{max}\,\mathsf{ide}\,N \cap (\mathsf{ide}\,P)^{\complement}$. By 3 we have $I \in \mathsf{max}\,I(M)$. But by $I \in (\mathsf{ide}\,P)^{\complement}$ there is $J \in \mathsf{ide}\,P$ with $I \subset J$, a contradiction to maximality of I. Hence $\mathsf{max}\,\mathsf{ide}\,N \cap (\mathsf{ide}\,P)^{\complement} = \emptyset$. By symmetry, also $\mathsf{max}\,\mathsf{ide}\,P \cap (\mathsf{ide}\,N)^{\complement} = \emptyset$. Now the claim is immediate from Lemma 30.2.4.
6. ($\subseteq$) follows from 6.
   ($\supseteq$) Assume $I \in \mathsf{max}\,\mathsf{ide}\,N \cap \mathsf{max}\,\mathsf{ide}\,P$. Then by 3 we have $I \in \mathsf{max}\,I(M)$. Hence, again by 3, we only need to show $I \in \mathsf{ide}\,(N \cap P)$. Since N and P are cones we get $I \subseteq N$ and $I \subseteq P$ and hence $I \subseteq N \cap P$ as well, showing the claim.

$\square$

The next lemma allows simplification of the defining property of a behaviour.

Lemma 12.4 *Consider $N, P \subseteq M$. Then*

$$\mathsf{max}\,\mathsf{ide}\,(N \cup P) = \mathsf{max}\,\mathsf{ide}\,P \Leftrightarrow \mathsf{ide}\,N \leq \mathsf{ide}\,P \ .$$

Proof. ($\Leftarrow$)

$\mathsf{ide}\,N \leq \mathsf{ide}\,P$

$\Rightarrow$ {[by Lemma 30.3.4]}

$\mathsf{max}\,(\mathsf{ide}\,N \cup \mathsf{ide}\,P) = \mathsf{max}\,\mathsf{ide}\,P$

$\Leftrightarrow$ $\qquad$ {[by Lemma 11.2]}

$\qquad \mathsf{max}\,\mathsf{ide}\,(N \cup P) \;=\; \mathsf{max}\,\mathsf{ide}\,P\;.$

($\Rightarrow$) If $N = \emptyset$, the claim holds trivially, since $\mathsf{ide}\,\emptyset = \emptyset$. Hence we now assume $N \neq \emptyset$.
We now need the so-called *Maximal Principle*, a variant of the Axiom of Choice (see eg. [19]): *Assume a partial order in which every non-empty chain has an upper bound. Then every element has a maximal element above it.*
We apply this to the partial order $(\mathsf{ide}\,N, \subseteq)$. It satisfies the assumption, since $\mathsf{ide}\,N$ is closed under directed unions and hence, in particular, under unions of chains. Consider now $I \in \mathsf{ide}\,N \subseteq \mathsf{ide}\,(N \cup P)$. By the maximal principle there is a $J \in \mathsf{max}\,\mathsf{ide}\,(N \cup P) = \mathsf{max}\,\mathsf{ide}\,P$ with $I \leq J$. □

Under additional assumptions we can simplify the assertion:

Lemma 12.5 *Assume $P \in \mathsf{dir}\,M$. Then*

$$\mathsf{max}\,\mathsf{ide}\,(N \cup P) \;=\; \mathsf{max}\,\mathsf{ide}\,P \;\Leftrightarrow\; N \leq P\;.$$

Proof. To apply Lemma 12.4 we show that $P \in \mathsf{dir}\,M$ implies

$$\mathsf{ide}\,N \leq \mathsf{ide}\,P \;\Leftrightarrow\; N \leq P\;.$$

($\Rightarrow$) Assume $x \in N$. Then $x^{\leq} \in \mathsf{ide}\,N$ and so there is $I \in \mathsf{ide}\,P$, say $I = D^{\leq}$ for $D \in \mathsf{dir}\,P$, with $x^{\leq} \leq I$. By Lemma 30.3.1-2 we get $x \leq P$.
($\Leftarrow$) For $I \in \mathsf{ide}\,N$ we have $I \leq P \in \mathsf{dir}\,P$ and hence, by Lemma 30.3.1, also $I \leq P^{\leq} \in \mathsf{ide}\,P$. □

For a counterexample when P is not directed see Example 14.2 in connection with Corollary 12.6.2 below.

Recalling the equivalence $\sim$ associated with the pre-order $\leq$, we obtain from the previous two lemmas

Corollary 12.6 *Consider $N, P \subseteq M$. Then*

1. $\mathsf{ide}\,N \sim \mathsf{ide}\,P \Rightarrow \mathsf{max}\,\mathsf{ide}\,N = \mathsf{max}\,\mathsf{ide}\,P$.
2. *If $N, P \in \mathsf{dir}\,M$ then*

$$N \sim P \;\Rightarrow\; \mathsf{max}\,\mathsf{ide}\,N \;=\; \mathsf{max}\,\mathsf{ide}\,P\;.$$

12.4 An Alternative Characterisation of Infinite Ideals

We conclude this section by an alternative characterisation of the set $\mathsf{inf}\,\mathsf{ide}\,P$ for property $P \subseteq M$. First we define

$$\mathsf{lim}\,P \;\stackrel{\text{def}}{=}\; \{I \in I(M) : I \cap P \in \mathsf{dir}\,M \;\wedge\; \mathsf{max}\,(I \cap P) \;=\; \emptyset\}\;.$$

This generalises the corresponding definition for infinite words or streams in [55, 60, 61, 65, 66] (to cite just a few), which is based on [20]. Other notations for $\mathsf{lim}\,P$ found in the literature are P^{δ} or $\boldsymbol{P}$. We can then show

Lemma 12.7 *1.* $\mathsf{inf}\,\mathsf{ide}\,P \subseteq \mathsf{lim}\,P$.
2. If $(M, \leq)$ is max*-determined then the reverse inclusion holds as well.*

Proof. We first note that

$$\begin{array}{ll} & I \in \mathsf{inf}\,\mathsf{ide}\,P \\ \Leftrightarrow & \quad \{\!\![\ \text{definition}\ ]\!\!\} \\ & I \in \mathsf{ide}\,P \wedge \mathsf{max}\,I = \emptyset \\ \Leftrightarrow & \quad \{\!\![\ \text{by Lemma 11.1}\ ]\!\!\} \\ & I = (I \cap P)^{\leq} \wedge \mathsf{max}\,I = \emptyset \\ \Leftrightarrow & \quad \{\!\![\ \text{equality}\ ]\!\!\} \\ & I = (I \cap P)^{\leq} \wedge \mathsf{max}\,(I \cap P)^{\leq} = \emptyset \\ \Leftrightarrow & \quad \{\!\![\ \text{Lemma 30.2.2}\ ]\!\!\} \\ & I = (I \cap P)^{\leq} \wedge \mathsf{max}\,(I \cap P) = \emptyset\ . \qquad (*) \end{array}$$

Now we prove our claims as follows:

1. $$\begin{array}{ll} & (*) \\ \Rightarrow & \quad \{\!\![\ \text{by Lemma 30.5.3}\ ]\!\!\} \\ & I \cap P \in \mathsf{dir}\,M \wedge \mathsf{max}\,(I \cap P) = \emptyset \\ \Leftrightarrow & \quad \{\!\![\ \text{definition}\ ]\!\!\} \\ & I \in \mathsf{lim}\,P\ . \end{array}$$
2. Let $(M, \leq)$ be max-determined and assume $I \in \mathsf{lim}\,P$. By $(*)$ it remains to show $I = (I \cap P)^{\leq}$. First, by monotonicity of downward closure we have $(I \cap P)^{\leq} \subseteq I^{\leq} = I$. Using Lemma 30.2.2 we obtain $\mathsf{max}\,(I \cap P)^{\leq} = \mathsf{max}\,(I \cap P) = \emptyset$, so that by max-determinedness $(I \cap P)^{\leq} \in \mathsf{max}\,I(M)$ and hence $(I \cap P)^{\leq} = I$.

□

12.5 About max-Determinedness

To investigate under which conditions a partial order is max-determined, we introduce some auxiliary notions. Let $F : \mathcal{P}(M) \to \mathcal{P}(M)$ be some function, such as dir or ide. We say that $N \subseteq M$ *has F-maxima* if every set in $F(N)$ has a maximal element. In addition to the functions mentioned we shall use

$$\begin{array}{rcl} \mathsf{ne}\,N & \overset{\text{def}}{=} & \{C \subseteq N : C \neq \emptyset\}\ , \\ \mathsf{chai}\,N & \overset{\text{def}}{=} & \{C \subseteq N : C \text{ non-empty chain}\}\ . \end{array}$$

Lemma 12.8 *If $N \subseteq M$ has* chai*-maxima, then it also has* ne*-maxima.*

Proof. Assume $\emptyset \neq D \subseteq N$ and $\mathsf{max}\, D = \emptyset$. Construct a chain $C \subseteq N$ as follows: Choose $x_0 \in D$ arbitrarily. Assume now that x_i has been found. Since $x_i \notin \emptyset = \mathsf{max}\, D$, there is $x_{i+1} \in D$ with $x_i < x_{i+1}$. Now for $C \overset{\text{def}}{=} \{x_i : i \in \mathbb{N}\}$ we have $\mathsf{max}\, C = \emptyset$, a contradiction. □

Corollary 12.9 *If $N \subseteq M$ has* chai*-maxima, then it also has* dir*-maxima.*

Proof. Every directed set is non-empty. □

We say that $(M, \leq)$ *separates ideals* if for all $I, J \in I(M)$ with $I \neq J$ the intersection $I \cap J$ has chai-maxima. The connection with max-determinedness is given by

Theorem 12.10 *$(M, \leq)$ is* max*-determined iff $(M, \leq)$ separates ideals.*

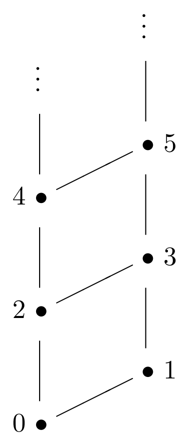
Proof. ($\Rightarrow$) Suppose $I \neq J$ and $C \in \mathsf{chai}\,(I \cap J)$, but $\mathsf{max}\, C = \emptyset$. Then $C^{\leq}$ is an ideal with $\mathsf{max}\, C^{\leq} = \emptyset$. By max-determinedness then $C^{\leq} \in \mathsf{max}\,\mathsf{ide}\, M$. Since by downward closedness of I and J we have $C^{\leq} \subseteq I$ and $C^{\leq} \subseteq J$ it follows that $I = C^{\leq} = J$, a contradiction.
($\Leftarrow$) Assume $\mathsf{max}\, I = \emptyset$ and $I \notin \mathsf{max}\,\mathsf{ide}\, M$. Then there is $J \neq I$ with $I \subseteq J$. Since $(M, \leq)$ separates ideals, and by Corollary 12.9, then $I = I \cap J$ has dir-maxima. In particular, $\mathsf{max}\, I \neq \emptyset$, a contradiction. □

This has the following surprising consequence:

Corollary 12.11 *Let $(M, \leq)$ be* max*-determined. Then all elements of M are compact.*

Proof. By the previous theorem, $(M, \leq)$ separates ideals.
We now first show $\sqcup I \subseteq I$ for all $I \in I(M)$. Assume $y \in \sqcup I$ and set $J \overset{\text{def}}{=} y^{\leq}$. We have $I \subseteq J$. If $I \neq J$ then $I = I \cap J$ has a maximal and hence, by directedness, greatest element z. But then $z = \sqcup I = y$ so that $J = I$, a contradiction.
Consider now $x \in M$ and $I \in I(M)$ such that $x \leq \sqcup I \in I$. By downward closedness of I we get $x \in I$ and x is compact. □

The reverse implication is not valid as the following example shows. Consider the partial order



in which all elements are compact. However, for $I \overset{\text{def}}{=} \{0, 2, 4, \ldots\}$ we have $\mathsf{max}\, I = \emptyset$ and $I \subset J \overset{\text{def}}{=} \{0, 1, 2, 3, 4, \ldots\}$, ie. I is not maximal. Concerning separation of ideals, $I = I \cap J$ doesn't have a maximal element.

It will be interesting to find further, more "manageable" characterisations of max-determinedness.

Part III: A Particular Case: Streams

We now specialize to a particular partial order. We shall represent streams using sets of finite traces. These are finite words over an alphabet A of atomic actions; they are ordered by the prefix relation.

13 Streams and Properties

Whenever we are working in the particular domain A^∞, we rename ide into str to emphasise that fact. So the set of streams spanned by property $P \subseteq A^*$ is

$$\mathsf{str}\, P \overset{\text{def}}{=} \mathsf{ide}\, P\ .$$

Note that it would not be adequate to work with the set $\mathsf{str}\,(P^{\sqsubseteq})$, the so-called *adherence* of P (see eg. [49, 60]), instead of $\mathsf{str}\, P$. The reason is that by prefix-closure infinite substreams may "sneak" into a cone although it results from a language of mutually $\sqsubseteq$-incomparable words which represent systems with finite behaviour only.

Example 13.1 The language $L \overset{\text{def}}{=} 0^* \bullet 1$ represents a behaviour with arbitrarily long but finite sequences of 0s terminated by the "explicit endmarker" 1. The words in L are mutually incomparable w.r.t. $\sqsubseteq$. Hence all directed subsets of L are singletons and their downward closures are principal ideals and hence finite. So $\mathsf{str}\,L$ consists of finite ideals only. However, the prefix closure $L^{\sqsubseteq}$ contains the infinite ideal 0^* representing the infinite stream 0^ω of 0s. So $\mathsf{str}\,L^{\sqsubseteq} = \mathsf{str}\,L \cup \{0^\omega\}$. $\square$

Using König's Lemma one can show that for finite A *every* infinite cone contains an infinite stream. The general definition of ide omits these undesired streams.

So, using ide we can distinguish between erratic and angelic non-determinacy and talk about fairness without resorting to metric and topological spaces as eg. in [4].

Example 13.2 Consider the recursive definition

$$\mathcal{B} = 0 \circ \mathcal{B} \,[]\, 1 ,$$

where $\circ$ denotes stream concatenation (see Section 15 for a precise definition) and $[]$ denotes non-deterministic choice. In an angelic interpretation of $[]$ always eventually the terminating branch 1 is chosen, and so $\mathcal{B}$ would equal $\mathsf{str}\,L$ of Example 13.1.

In an erratic interpretation of $[]$, on the other hand, no guarantee is given that the terminating branch will ever be chosen, and so $\mathcal{B}$ would equal $\mathsf{str}\,L^{\sqsubseteq}$ of Example 13.1. $\square$

We want to show now that str (and hence ide) does not distribute through general union:

Example 13.3 Take $U = 0^*$. Then $U = \bigcup_{i \in \mathbb{N}} 0^i$. However, $\mathsf{str}\,U = \{0^*\} \cup \{(0^i)^{\sqsubseteq} : i \in \mathbb{N}\}$, whereas $\bigcup_{i \in \mathbb{N}} \mathsf{str}\,0^i = \{(0^i)^{\sqsubseteq} : i \in \mathbb{N}\}$. $\square$

14 Maximal and Infinite Streams

As already mentioned, maximal ideals model processes that go on as long as possible. For streams we have a more pleasant situation than for general ideals:

Lemma 14.1 *$(A^*, \sqsubseteq)$ is max-determined.*

Proof. Assume $I \in \mathsf{ide}\,A^* \wedge \mathsf{max}\,I = \emptyset$ and consider $J \in \mathsf{ide}\,A^*$ with $I \subseteq J$. By Lemma 30.4 and downward closure of I, J it suffices to show $J \leq I$. Consider $y \in J$. Since $\mathsf{max}\,I = \emptyset$ there is some $x \in I \subseteq J$ with $||y|| \leq ||x||$, where $||u||$ denotes the length of word u. Moreover, by directedness of J, there is $z \in J$ with $x \sqsubseteq z \wedge y \sqsubseteq z$. From linearity of $z^{\sqsubseteq}$ it therefore follows that $x \sqsubseteq y \vee y \sqsubseteq x$. However, since $||y|| \leq ||x||$, we must have $y \sqsubseteq x$. $\square$

This allows us to use all laws from Section 12 for streams. At this point it is also convenient to give the counterexample to the simplified version of Corollary 12.6:

Example 14.2 Set $U \overset{\text{def}}{=} 0^* \bullet 1$ and $V \overset{\text{def}}{=} U \cup 0^* = U^{\sqsubseteq}$ by (3). Then $U \sim V$, but $\mathsf{max}\,\mathsf{str}\,U \neq \mathsf{max}\,\mathsf{str}\,V$, since $0^* \in (\mathsf{max}\,\mathsf{str}\,V)\backslash(\mathsf{max}\,\mathsf{str}\,U)$. □

Concerning infinite streams, we note that by Lemma 8.2 we have

$$\mathsf{inf}\,\mathsf{ide}\,P = \{I \in \mathsf{ide}\,P : I \text{ infinite}\} .$$

To establish the relation with [60] we also show

Lemma 14.3 *For $P \subseteq A^*$ we have*

$$\mathsf{lim}\,P = \{I \in A^\infty : I \cap P \textit{ infinite}\} .$$

Proof. $I \in \mathsf{lim}\,P$

$\Leftrightarrow$ $\{\!\![$ definition $]\!\!\}$

$I \cap P \in \mathsf{dir}\,M \wedge \mathsf{max}\,(I \cap P) = \emptyset$

$\Leftrightarrow$ $\{\!\![$ by $I \cap P \subseteq I$ and Lemma 8.1 $]\!\!\}$

$I \cap P \neq \emptyset \wedge \mathsf{max}\,(I \cap P) = \emptyset$.

We show now that, for linearly ordered $L \subseteq A^*$,

$$L \text{ infinite } \Leftrightarrow L \neq \emptyset \wedge \mathsf{max}\,L = \emptyset .$$

($\Rightarrow$) $L \neq \emptyset$ is immediate. Suppose $x \in \mathsf{max}\,L$. By linearity then $L \subseteq x^{\sqsubseteq}$. But then $|L| \leq ||x|| + 1$, a contradiction.
($\Leftarrow$) Every non-empty finite set has a maximal element. □

To end this section, we write out specialisations of some of our laws for the case of streams, since they will be used in the bounded buffer example below:

Corollary 14.4

$$\begin{aligned} \mathsf{inf}\,\mathsf{str}\,(N \cup P) = \mathsf{inf}\,\mathsf{str}\,P &\Leftarrow N \sqsubseteq P \wedge P \textit{ directed} \\ \mathsf{inf}\,\mathsf{str}\,N = \mathsf{inf}\,\mathsf{str}\,P &\Leftarrow N \sim P \wedge N, P \textit{ directed} \end{aligned}$$

Proof. Immediate from Lemma 14.1, Lemma 12.5 and Corollary 12.6. □

Example 14.5 Since $(a \bullet b)^* \bullet a) \sim (a \bullet b)^*$ and both languages are directed, we obtain $(\mathsf{inf}\,\mathsf{str}\,(a \bullet b)^* \bullet a) = \mathsf{inf}\,\mathsf{str}\,(a \bullet b)^*$. □

15 Stream Concatenation

As a prerequisite for defining infinite repetition we need stream concatenation which, for streams S, T is defined by

$$S \circ T \overset{\text{def}}{=} S \cup (\mathsf{max}\, S) \bullet T \ .$$

Let us explain this definition. If S is finite then $\mathsf{max}\, S$ is a singleton. This part of the overall behaviour then is prefixed to all traces in T to represent the concatenated behaviour. If S is infinite then $\mathsf{max}\, S = \emptyset$ and hence, by strictness of $\circ$, we get $S \circ T = S$, as is intuitively expected. We have

$$\mathsf{max}\,(S \circ T) \ = \ (\mathsf{max}\, S) \bullet (\mathsf{max}\, T) \ .$$

It is straightforward to show that $S \circ T$ is indeed a stream and that $(A^\infty, \circ, \varepsilon)$ is a monoid. As a shorthand notation we shall also allow words as first argument of $\circ$. This is made precise by setting

$$u \circ T \ \overset{\text{def}}{=} \ u^{\sqsubseteq} \circ T \ = \ u^{\sqsubseteq} \cup u \bullet T \ .$$

Again, $\circ$ is extended pointwise to behaviours and, in the case of the above shorthand, to languages.

16 Infinite Repetition

We now give the usual greatest fixpoint definition of the set U^ω of streams that result from infinite repetition of words from a language $U \subseteq A^*$:

$$\begin{array}{l} U^\omega \ = U \circ U^\omega \ \wedge \\ \mathcal{X} \ = U \circ \mathcal{X} \ \Rightarrow \ \mathcal{X} \subseteq U^\omega \ . \end{array}$$

According to the Knaster-Tarski fixpoint theorem this is well-defined by monotonicity of $\circ$. Note that by this definition $\emptyset^\omega = \emptyset$. However, if $\varepsilon \in U$ then $U^\omega = A^\infty$. For that reason, U^ω is usually considered only for $\varepsilon \notin U$.

It should be noted that for $|U| \geq 2$ and $\varepsilon \notin U$ there are nontrivial solutions of $\mathcal{X} = U \circ \mathcal{X}$ properly less than U^ω. As an example consider the behaviour $U^* \circ \bigcup\limits_{u \in U} u^\omega$ of all eventually periodic streams.

To tie this in with the str-operation, we quote [60], p. 433:

$$\varepsilon \notin U \ \Rightarrow \ \mathsf{lim}\, U^* \ = \ U^\omega \ \cup \ U^* \circ \mathsf{lim}\, U \ ,$$

or, using Lemma 12.7 and max -determinedness,

$$\varepsilon \notin U \ \Rightarrow \ \mathsf{inf\,str}\, U^* \ = \ U^\omega \ \cup \ U^* \circ \mathsf{inf\,str}\, U \ .$$

From this, by strictness of $\circ$ it is immediate that

$$\varepsilon \notin U \ \wedge \ \mathsf{inf\,str}\, U \ = \ \emptyset \ \ \Rightarrow \ \ U^\omega \ = \ \mathsf{inf\,str}\, U^* \ . \qquad (6)$$

A sufficient condition to establish the premise is given by

Lemma 16.1 *If $U \subseteq A^*\backslash\varepsilon$ satisfies the Fano condition, ie. the words in U are mutually incomparable w.r.t. $\sqsubseteq$, then*

$$U^\omega \;=\; \mathsf{inf}\,\mathsf{str}\,U^* \;.$$

Proof. By the Fano condition, all directed subsets of U are singletons. Hence $\mathsf{str}\,U \;=\; \{u^{\leq} : u \in U\}$ consists of finite streams only. $\square$

Note that if $\varepsilon \in U$ then U satisfies the Fano condition iff $U \;=\; \varepsilon$; for this case the above equation doesn't hold, since then $\mathsf{inf}\,\mathsf{str}\,U^* \;=\; \emptyset$. It should also be mentioned that U satisfies the Fano condition iff $U \;=\; \mathsf{max}\,U$. To see what happens if the Fano condition is not satisfied, consider

Example 16.2 Let $A = \{a, b\}$ and $U \overset{\text{def}}{=} \{a \bullet b^n : n \in \mathbb{N}\} \subseteq A^*$. Then $U \in \mathsf{dir}\,U^*$, since $U \subseteq U^*$ and U is directed. Hence $U^{\sqsubseteq} = \varepsilon \cup U \in \mathsf{str}\,U^*$ and, since $U^{\sqsubseteq}$ is infinite, even $U^{\sqsubseteq} \in \mathsf{inf}\,\mathsf{str}\,U^*$. Now, $U^{\sqsubseteq}$ represents an a followed by infinitely many bs; but this behaviour clearly does not arise from repeated concatenation of words in U. It is "sneaked in" by the fact that simply considering directed subsets of U^* throws away too much structural information. $\square$

To allow a characterisation of U^ω for languages that do not satisfy the Fano condition, one can artificially enforce it by attaching a special endmarker to all words in U and remove it after singling out the infinite streams. Let $\# \notin A$ be a new letter and consider streams over the extended alphabet $A \cup \#$. Moreover, denote by $A \lhd u$ the word that results from u by removing all occurrences of $\#$ and extend the operation $A\lhd$ pointwise to languages and behaviours. Then we have

Lemma 16.3 *For $U \subseteq A^*\backslash\varepsilon$,*

$$U^\omega \overset{\text{def}}{=} A \lhd \mathsf{inf}\,\mathsf{str}\,(U \bullet \#)^* \;.$$

For the somewhat tedious proof see [43].

The streams in $\mathsf{str}\,(U \bullet \#)^*$ correspond to finite and infinite sequences that result from concatenating arbitrary elements of U with the separator $\#$ in between. The operation inf then selects the infinite ones of these; if $\varepsilon \notin U$ these are precisely the infinite words resulting from repeatedly concatenating words from U. The separators are used to record the "construction history" of the streams; they are finally thrown away again by the filter $A\lhd$. In this way subsets of U^* which are directed "by accident" are ignored. A similar mechanism for defining iteration is employed in [50] in the finite case and in [11] in the infinite case.

17 Streams of Functions

We have made no assumptions about our alphabet A. Hence it may even be a set of functions. Then streams over A model components with time-dependent behaviour. We have seen examples of this in the description of various faulty channels in Section 5.

A stream $S \in A^\infty$ of arguments is fed into a stream $F \in (A \to A^*)^\infty$ of functions by the operator $>>$. The images $f(a)$ of the elements a of A under the individual functions f in F are concatenated into an overall output stream. A wordwise definition of this is

$$\begin{aligned} \varepsilon >> w &\overset{\text{def}}{=} \varepsilon \ , \\ s >> \varepsilon &\overset{\text{def}}{=} \varepsilon \ , \\ a \bullet s >> f \bullet w &\overset{\text{def}}{=} f(a) \bullet (s >> w) \ . \end{aligned}$$

This operation is extended pointwise to languages and behaviours.

Example 17.1 For finite stream S we have

$$(S \bullet T) >> \mathit{cchan}_* \ = \ (S >> \mathit{arbchan}) \bullet (T >> \mathit{cchan}_*) \ .$$

This reflects the unbounded fairness of cchan_*: we have no guarantee *when* correct transmission occurs, and hence the elements of S may or may not be transmitted correctly. □

With bound assumptions one gets more precise information:

Example 17.2 We have

$$m > k \ \Rightarrow \ (a^m \bullet T) >> \mathit{cchan}_{\leq k} \ \in \ A^{\leq k} \bullet a \bullet A^\infty \ .$$

A channel with fairness bound k *must* transmit a correctly at least once if it receives more than k copies of a. □

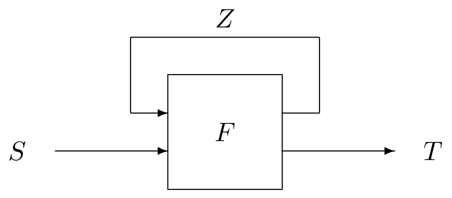
18 Feedback and State-Based Systems

18.1 The Feedback Operation

An essential operation on SPFs is *feedback* of some outputs to the inputs. Assume an SPF $F : A^\infty \times B^\infty \to A^\infty \times C^\infty$. Then its feedback $\mathit{feed}F : B^\infty \to C^\infty$ is given by

$$(\mathit{feed}F)(S) \ = \ T \,\mathsf{where}\, (Z, T) = F(Z, S) \ .$$

It may be depicted as



The semantics of this recursive declaration is the usual least-fixpoint one. This version of the feedback operator hides the feedback stream. If this is to made visible one simply copies it and feeds one copy back whereas the other is transmitted to the outside.

18.2 State-Based Systems and Automata

This operation together with streams of functions allows a very convenient and concise description of state-based systems.

Assume a set Q of states, an input alphabet A and an output alphabet B. Then a *time-dependent automaton* is given by a stream $H \in (Q \times A \to Q \times B)^\infty$.

We may now feed this automaton with a starting state $q_0 \in Q$ and a stream $S \in A^\infty$ of input values to produce a stream of output values in B^∞. The stream of states entered during the processing of the input is constructed by a feedback and hidden from the outside. This is described by

$$auto(H, q_0, S) = T \,\mathsf{where}\, (Z, T) = (q_0 \bullet Z, S) >> H \ .$$

By placing various restrictions on the entities involved, we can distinguish a hierarchy of automata:

– If no further restrictions are made, we obtain a *timed and state-dependent* automaton.
– If we require $|Q| = 1$ then we have a *timed and state-independent* automaton.
– If we take $H = f^\omega$ for some $f : Q \times A \to Q \times B$, we obtain a *timeless and state-dependent* automaton.
– If we again take $H = f^\omega$ but also require $|Q| = 1$, we have a *timeless and state-independent* automaton.

For example, an easy proof by induction over the structure of the finite words shows

Lemma 18.1 *If* $|Q| = 1$ *then*

$$auto(f^\omega, q_0, S) \ = \ S >> g^\omega \ ,$$

where $g(i) \ = \ \pi_2(f(q_0, i))$.

We now illustrate the general case by the following

Example 18.2 We give a description of a one-place asynchronous buffer. The example is taken from [10]. Consider a set D of data. The input alphabet is

$$A \overset{\text{def}}{=} D \cup \{!\} \ .$$

An input $d \in D$ means that d is to be stored in the buffer, whereas ! means a request for the current contents of the buffer.

At each time point the buffer may accept or reject its input which is shown by a Boolean value. In addition to that the buffer will output data if it accepts the request signal. So we choose the output alphabet

$$B \overset{\text{def}}{=} (D \cup \{\varepsilon\}) \times \mathbb{B} \ ,$$

where ε models the case of no proper output.

As the set of states we choose

$$Q \overset{\text{def}}{=} D \cup \{\varepsilon\}$$

where ε models the state of being empty whereas $d \in D$ models the state of containing value d.

Now we define two transition functions

$$acc, rej : Q \times A \to Q \times B$$

which model acceptance and rejection of the input. We have

$$\begin{aligned} acc(q, d) &= (\mathsf{if}\ q = \varepsilon\,\mathsf{then}\,d\,\mathsf{else}\,q, (\varepsilon, q = \varepsilon))\ , \\ acc(q, !) &= (\varepsilon, (q, q \neq \varepsilon))\ , \\ rej(q, x) &= (q, (\varepsilon, \mathsf{false}))\ . \end{aligned}$$

The behaviour of a fair buffer, ie. one which rejects inputs only finitely many times before eventually accepting one is the specified as

$$auto((rej^* \bullet acc)^\omega, \varepsilon)\ .$$

In particular, we can avoid the use of prophecy variables (see eg. [10]) in this style. □

19 Processes and Synchronised Parallel Composition

While the previous two sections are appropriate for the SPF view of distributed systems, we now define operators that are adequate for the trace view (cf. Section 6). The particular definitions given here draw strongly on the corresponding ones in [25].

Assume an overall alphabet A for our streams. A *process* is a pair $(B, \mathcal{B})$ where $B \subseteq A$ is the *alphabet* of the process and $\mathcal{B} \subseteq B^\infty$ is a behaviour. We set

$$\alpha(B, \mathcal{B}) \overset{\text{def}}{=} B\ , \qquad \beta(B, \mathcal{B}) \overset{\text{def}}{=} \mathcal{B}\ .$$

An auxiliary operation is the *projection* † of words to an alphabet $B \subseteq A$. It is defined inductively as follows:

$$\begin{aligned} \varepsilon \dagger B &\overset{\text{def}}{=} \varepsilon \\ (a \bullet s) \dagger B &\overset{\text{def}}{=} \begin{cases} a \bullet (s \dagger B) & \text{if } a \in B \\ s \dagger B & \text{otherwise.} \end{cases} \end{aligned}$$

Projection is extended pointwise to languages and behaviours. The projection of a stream is a stream again.

Using projection we can characterise processes in another way: the pair $(B, \mathcal{B})$ is a process iff $\forall\ S \in \mathcal{B} : S \dagger B = S$.

We need to lift the notion of refinement to processes. We allow that a process is refined by another one that has additional “internal” actions. Since then refinement amounts to inclusion of (the projection of) the behaviour, we abuse notation and write again $\subseteq$ for the refinement relation:

$$P \subseteq Q \overset{\text{def}}{\Leftrightarrow} \alpha P \supseteq \alpha Q \ \wedge \ (\beta P) \dagger \alpha Q \subseteq \beta Q \ .$$

In this case we say that P *refines* Q. It is easily checked that $\subseteq$ is a partial order on processes.

If behaviours are "loose enough" in that they allow arbitrary actions in between the "interesting" ones, one can model synchronised parallel composition very simply by intersection (see eg. [26]). For general behaviours this works well only if they are "loosened" by interspersing arbitrary actions between the proper ones; this is again taken from [25]. The intersection then allows only traces in which the actions interesting to both partners occur in a sequence that is acceptable to both partners (ie. allowed in both behaviours) whereas the private actions of each partner are not constrained by the other partner.

Hence, for processes P and Q, we define the parallel composition $P||Q$ by

$$\begin{aligned} \alpha(P||Q) &\overset{\text{def}}{=} \alpha P \cup \alpha Q \ , \\ S \in \beta(P||Q) &\overset{\text{def}}{\Leftrightarrow} S = S \dagger \alpha(P||Q) \wedge \\ &\ \ S \dagger \alpha P \in \beta P \wedge \\ &\ \ S \dagger \alpha Q \in \beta Q \ . \end{aligned}$$

Note, in particular, that $||$ is commutative, associative and idempotent then. Moreover,

$$P \subseteq Q \ \Leftrightarrow \ P||Q = P \ .$$

If $\alpha P = \alpha Q$ then $\beta(P||Q) = \beta P \cap \beta Q$.

This parallel composition operator will be used in our extended example in Section 22.3.

Part IV: Safety and Liveness

We have already informally discussed safety and liveness (see eg. [33, 1, 21]). We want to show how these notions can be expressed algebraically. In [1] and subsequent papers, a *property* is a set of infinite sequences of states. The appropriate counterpart in our setting is therefore a set of streams, more generally, ideals, ie. a *behaviour*.

20 Safety

20.1 Definition and Topological Properties

In [1] a behaviour $\mathcal{B} \subseteq A^\omega$ over infinite streams is called *safe* if the following holds:

$$\forall\, S \in A^\omega : S \notin \mathcal{B} \Rightarrow (\exists\, s \in S : \forall\, T \in A^\omega : s \circ T \notin \mathcal{B}) \ .$$

This means that for every stream not in the behaviour there is a decisive finite prefix s where something went "irreparably wrong" in that *no* continuation of s can bring the computation back to the "good path".

We want to simplify the formal definition above by moving from logic to algebra. First, using contraposition, the formula can be transformed to

$$\forall\ S \in A^\omega : (\forall\ s \in S : \exists\ T \in A^\omega : s \circ T \in \mathcal{B}) \ \Rightarrow\ S \in \mathcal{B}\ .$$

Now, recalling the definition $\mathsf{pref}\,\mathcal{B} = \cup\mathcal{B}$ from Section 10, we have

$$\exists\ T \in A^\omega : s \circ T \in \mathcal{B} \ \Leftrightarrow\ s \in \mathsf{pref}\,\mathcal{B}\ . \tag{7}$$

Hence the safety condition reduces to

$$\begin{array}{ll} & \forall\ S \in A^\omega : (\forall\ s \in S : s \in \mathsf{pref}\,\mathcal{B}) \ \Rightarrow\ S \in \mathcal{B} \\ \Leftrightarrow & \quad \{\![\ \text{set theory}\ ]\!\} \\ & \forall\ S \in A^\omega : S \subseteq \mathsf{pref}\,\mathcal{B} \ \Rightarrow\ S \in \mathcal{B} \\ \Leftrightarrow & \quad \{\![\ \text{by prefix-closedness of } \mathsf{pref}\,\mathcal{B} \text{ and Lemma 11.4.1}\ ]\!\} \\ & \forall\ S \in A^\omega : S \in \mathsf{str}\,\mathsf{pref}\,\mathcal{B} \ \Rightarrow\ S \in \mathcal{B} \\ \Leftrightarrow & \quad \{\![\ \text{defining } \mathsf{should}\,\mathcal{B} \overset{\text{def}}{=} \mathsf{str}\,\mathsf{pref}\,\mathcal{B}\ ]\!\} \\ & \mathsf{should}\,\mathcal{B} \subseteq \mathcal{B}\ . \end{array}$$

This simplified form involves only order-theoretic notions and hence generalises easily to arbitrary ideal completions. Consider a partial order $(M, \leq)$ and a behaviour $\mathcal{B} \subseteq M^\infty$. Then we call $\mathcal{B}$ *safe* iff $\mathsf{should}\,\mathcal{B} \subseteq \mathcal{B}$, where

$$\mathsf{should}\,\mathcal{B} \overset{\text{def}}{=} \mathsf{ide}\,\mathsf{pref}\,\mathcal{B}\ .$$

By $\subseteq$-monotonicity of pref and ide also should is $\subseteq$-monotonic. Note that for all $\mathcal{B} \subseteq M^\infty$ we have $\mathcal{B} \subseteq \mathsf{should}\,\mathcal{B}$. So a behaviour $\mathcal{B} \subseteq M^\infty$ is safe iff $\mathcal{B} = \mathsf{should}\,\mathcal{B}$. Moreover,

Lemma 20.1 *1. Safe behaviours are closed under arbitrary intersections and finite unions.*
2. should *is idempotent.*
3. $\mathsf{should}\,\mathcal{B}$ *is the least safe behaviour containing* $\mathcal{B}$.

Proof. 1. Assume a family $(\mathcal{B}_j)_{j\in J}$ of safe behaviours. Then for all $j \in J$ we have by monotonicity of should and safety of $\mathcal{B}_j$ that

$$\mathsf{should}\,(\bigcap_{j\in J} \mathcal{B}_j) \subseteq \mathsf{should}\,\mathcal{B}_j \subseteq \mathcal{B}_j\ ,$$

so that

$$\mathsf{should}\,(\bigcap_{j\in J} \mathcal{B}_j) \subseteq \bigcap_{j\in J} \mathcal{B}_j\ ,$$

ie. $\bigcap_{j\in J} \mathcal{B}_j$ is safe again.

For union we calculate, for $I \in M^\infty$,

$$
\begin{array}{cl}
 & I \in \mathsf{should}\,(\mathcal{B} \cup \mathcal{C}) \\
\Leftrightarrow & \quad \{\!\![\text{ definition and distributivity of } \mathsf{pref} \ ]\!\!\} \\
 & I \subseteq \mathsf{pref}\,\mathcal{B} \cup \mathsf{pref}\,\mathcal{C} \\
\Leftrightarrow & \quad \{\!\![\text{ Boolean algebra }]\!\!\} \\
 & I = (I \cap \mathsf{pref}\,\mathcal{B}) \cup (I \cap \mathsf{pref}\,\mathcal{C}) \ .
\end{array}
$$

Set now $I_{\mathcal{B}} \stackrel{\text{def}}{=} I \cap \mathsf{pref}\,\mathcal{B}$ and $I_{\mathcal{C}} \stackrel{\text{def}}{=} I \cap \mathsf{pref}\,\mathcal{C}$. Since I is directed, by Lemma 30.5.2 we have $I_{\mathcal{B}} \leq I_{\mathcal{C}}$ or $I_{\mathcal{C}} \leq I_{\mathcal{B}}$. So by downward closedness of $I_{\mathcal{B}}$ and $I_{\mathcal{C}}$ and Lemma 30.3.3 we have $I_{\mathcal{B}} \subseteq I_{\mathcal{C}}$ or $I_{\mathcal{C}} \subseteq I_{\mathcal{B}}$ and hence $I = I_{\mathcal{B}}$ or $I = I_{\mathcal{C}}$. But then, by Boolean algebra and the definition, we get $I \in \mathsf{should}\,\mathcal{B} \vee I \in \mathsf{should}\,\mathcal{C}$, so that by safety of $\mathcal{B}$ and $\mathcal{C}$ also $I \in \mathcal{B} \cup \mathcal{C}$.

2. For $I \in M^\infty$ we have

$$
\begin{array}{cl}
 & I \in \mathsf{should}\,\mathsf{should}\,\mathcal{B} \\
\Leftrightarrow & \quad \{\!\![\text{ definitions }]\!\!\} \\
 & I \subseteq \bigcup\{J \in M^\infty : J \subseteq \mathsf{pref}\,\mathcal{B}\} \\
\Leftrightarrow & \quad \{\!\![\text{ using the principal ideals } J = x^{\leq} \text{ for } x \in \mathsf{pref}\,\mathcal{B} \ ]\!\!\} \\
 & I \subseteq \mathsf{pref}\,\mathcal{B} \\
\Leftrightarrow & \quad \{\!\![\text{ definitions }]\!\!\} \\
 & I \in \mathsf{should}\,\mathcal{B} \ .
\end{array}
$$

3. Let $\mathcal{C}$ be safe with $\mathcal{B} \subseteq \mathcal{C}$. Then

$$
\begin{array}{cl}
 & \mathsf{should}\,\mathcal{B} \\
\subseteq & \quad \{\!\![\text{ monotonicity }]\!\!\} \\
 & \mathsf{should}\,\mathcal{C} \\
= & \quad \{\!\![\text{ safety of } \mathcal{C} \ ]\!\!\} \\
 & \mathcal{C} \ .
\end{array}
$$

But $\mathsf{should}\,\mathcal{B}$ is safe by 2.

□

By these properties, the safe behaviours coincide with the closed sets of a topology on M^∞ (cf. eg. [59]) and should is the topological closure operator.

20.2 Safety and Snapshot sets

Let us now study how safety is reflected in snapshot sets. In other words, we want to know when for $P \subseteq M$ the behaviour $\mathsf{ide}\,P$ is safe. We calculate, for $I \in M^\infty$,

$I \in \mathsf{should}\,\mathsf{ide}\,P$

$\Leftrightarrow$ $\{\![$ definition of should $]\!\}$

$I \in \mathsf{ide}\,\mathsf{pref}\,\mathsf{ide}\,P$

$\Leftrightarrow$ $\{\![$ by Lemma 11.4.2 $]\!\}$

$I \in \mathsf{ide}\,(P^{\leq})$

$\Leftrightarrow$ $\{\![$ by Lemma 11.4.1 $]\!\}$

$I \subseteq P^{\leq}$

and hence

$\mathsf{ide}\,P$ is safe

$\Leftrightarrow$ $\{\![$ by the above $]\!\}$

$\forall\, I \in M^{\infty} : I \subseteq P^{\leq} \Rightarrow I \in \mathsf{ide}\,P$

$\Rightarrow$ $\{\![\ \forall\, u \in P : u^{\leq} \subseteq P^{\leq}\ ]\!\}$

$\forall\, u \in P^{\leq} : u^{\leq} \in \mathsf{ide}\,P$

$\Rightarrow$ $\{\![$ since for $D \in \mathsf{dir}\,P$ we have $D^{\leq} = u^{\leq} \Leftrightarrow u \in D$ $]\!\}$

$P^{\leq} \subseteq P$.

On the other hand,

$$P^{\leq} \subseteq P \Rightarrow \forall\, I \in M^{\infty} : I \subseteq P^{\leq} \Rightarrow I \subseteq P\ .$$

Altogether we have shown

Lemma 20.2 *The behaviour* $\mathsf{ide}\,P$ *is safe iff* $P^{\leq} \subseteq P$*, ie. iff* P *is downward closed.*

For that reason we call a snapshot set $P \subseteq M$ a *safety property* iff it is downward closed. We have

Corollary 20.3 *If* $I \in M^{\infty}$ *and* P *is a safety property, then*

$$I \in \mathsf{ide}\,P \Leftrightarrow I \subseteq P\ .$$

Proof. Immediate from Lemma 11.4.1. □

For a safety property P the behaviour $\mathsf{ide}\,P$ is closed under unions (ie. suprema) of $\subseteq$-ascending chains of streams. In the special case of streams, safety properties are simply prefix-closed subsets of A^*.

21 Continual Satisfaction

21.1 The General Case

In connection with safety issues one is interested in the set of all objects that satisfy a property also in all their finite approximations. Given a property $P \subseteq M$ we define the property $\mathsf{saf}\, P$ by

$$\mathsf{saf}\, P \overset{\text{def}}{=} \{x \in M : x^{\leq} \subseteq P\} \ .$$

The set $\mathsf{saf}\, P$ has also been termed the *prefix kernel* of P in [50, 67]. We have

Lemma 21.1 *1.* $\mathsf{saf}\, P \subseteq P$.
2. $\mathsf{saf}\, P = P$ *iff* P *is a safety property. In particular,* $\mathsf{saf}\, P^{\leq} = P^{\leq}$.
3. $\mathsf{saf}\, P$ *is the greatest safety property contained in* P.
4. saf *is monotonic and strict w.r.t.* $\emptyset$.
5. $\mathsf{saf}\,(P \cap Q) = \mathsf{saf}\, P \cap \mathsf{saf}\, Q$.
6. $I \in \mathsf{ide}\,\mathsf{saf}\, P \Leftrightarrow I \subseteq P$.

Proof. 1. $x \in \mathsf{saf}\, P$

$\Leftrightarrow$ $\{\![$ definition $]\!\}$

$x^{\leq} \subseteq P$

$\Rightarrow$ $\{\![$ $x \in x^{\leq}$ $]\!\}$

$x \in P$.

2. $(\Rightarrow)$

$x \in P$

$\Leftrightarrow$ $\{\![$ assumption $]\!\}$

$x \in \mathsf{saf}\, P$

$\Leftrightarrow$ $\{\![$ definition $]\!\}$

$x^{\leq} \subseteq P$.

$(\Leftarrow)$

$x \in P$

$\Rightarrow$ $\{\![$ assumption $]\!\}$

$x^{\leq} \subseteq P$

$\Leftrightarrow$ $\{\![$ definition $]\!\}$

$x \in \mathsf{saf}\, P$

so $P \subseteq \mathsf{saf}\, P$; the reverse inclusion was shown in 1.

3. It is obvious that $\mathsf{saf}\, P$ is a safety property. Let $Q \subseteq P$ be a safety property and $x \in Q$. By definition then $x^{\leq} \subseteq Q \subseteq P$ and hence $x \in \mathsf{saf}\, P$.

4. Immediate from the definition.
5. $$\begin{array}{ll} & x \in \mathsf{saf}\,(P \cap Q) \\ \Leftrightarrow & \quad \{\!\![\text{ definition }]\!\!\} \\ & x^{\leq} \subseteq P \cap Q \\ \Leftrightarrow & \quad \{\!\![\text{ infimum property of intersection }]\!\!\} \\ & x^{\leq} \subseteq P \wedge x^{\leq} \subseteq Q \\ \Leftrightarrow & \quad \{\!\![\text{ definition }]\!\!\} \\ & x \in \mathsf{saf}\,P \wedge x \in \mathsf{saf}\,Q\ . \end{array}$$
6. $$\begin{array}{ll} & I \in \mathsf{ide}\,\mathsf{saf}\,P \\ \Leftrightarrow & \quad \{\!\![\text{ by Lemma 11.1 }]\!\!\} \\ & I \subseteq (I \cap \mathsf{saf}\,P)^{\leq} \\ \Leftrightarrow & \quad \{\!\![\text{ since } \mathsf{saf}\,P \subseteq P \text{ }]\!\!\} \\ & I \subseteq \mathsf{saf}\,P^{\leq} \\ \Leftrightarrow & \quad \{\!\![\text{ by downward closedness of } \mathsf{saf}\,P \text{ }]\!\!\} \\ & I \subseteq \mathsf{saf}\,P \\ \Leftrightarrow & \quad \{\!\![\text{ by downward closedness of } I \text{ }]\!\!\} \\ & I \subseteq P\ . \end{array}$$

□

Note that saf does not distribute through union. We can now state further distributivity properties for ide:

Lemma 21.2 *Consider $N, P \subseteq M$. Then*

1. $N = \mathsf{saf}\,N \Rightarrow \mathsf{ide}\,(N \cap P) = \mathsf{ide}\,N \cap \mathsf{ide}\,P$.
2. $N = \mathsf{saf}\,N \Rightarrow \mathsf{inf}\,\mathsf{ide}\,(N \cap P) = \mathsf{inf}\,\mathsf{ide}\,N \cap \mathsf{inf}\,\mathsf{ide}\,P$.
3. $\mathsf{ide}\,Q \cap \mathsf{ide}\,P \subseteq \mathsf{ide}\,(Q^{\leq} \cap P^{\leq})$.

Proof. 1. We only need to show $(\supseteq)$, since the reverse inclusion follows from monotonicity of ide.
Assume $S \in \mathsf{ide}\,N \cap \mathsf{ide}\,P$, say $S = D^{\leq} = E^{\leq}$ with $D \in \mathsf{dir}\,N \wedge E \in \mathsf{dir}\,P$. By Lemma 30.4 then $E \leq D$, and by Lemma 30.3.2 we get $E \leq N$, since $D \subseteq N$. Now $N = N^{\leq}$ shows $E \subseteq N$. Since $E \subseteq P$ we get $E \subseteq N \cap P$ and, since E is directed, even $E \in \mathsf{dir}\,(N \cap P)$. This shows that $S = E^{\leq}$ and hence $S \in \mathsf{ide}\,(N \cap P)$.
2. immediate from 2 and equation (5).
3. Immediate from 1, Lemma 21.1.2 and monotonicity of ide.

□

21.2 Deriving a Recursion for saf

Next, for the particular case of streams we want to derive a grammar-like or automaton-like representation for safety properties of the form $\mathsf{saf}\,P$ for some $P \subseteq A^*$. We use induction on the words involved. For the induction base we calculate

$$
\begin{array}{ll}
 & \varepsilon \in \mathsf{saf}\,P \\
\Leftrightarrow & \quad \{\!\![\ \text{definition}\ ]\!\!\} \\
 & \varepsilon^{\sqsubseteq} \subseteq P \\
\Leftrightarrow & \quad \{\!\![\ \varepsilon^{\sqsubseteq} = \varepsilon\ ]\!\!\} \\
 & \varepsilon \in P\ .
\end{array}
$$

For the induction step, we have, for arbitrary $c \in A$,

$$
\begin{array}{ll}
 & c \bullet s \in \mathsf{saf}\,P \\
\Leftrightarrow & \quad \{\!\![\ \text{definition}\ ]\!\!\} \\
 & (c \bullet s)^{\sqsubseteq} \subseteq P \\
\Leftrightarrow & \quad \{\!\![\ \text{by (3)}\ ]\!\!\} \\
 & c^{\sqsubseteq} \cup c \bullet s^{\sqsubseteq} \subseteq P \\
\Leftrightarrow & \quad \{\!\![\ \text{set theory}\ ]\!\!\} \\
 & c^{\sqsubseteq} \subseteq P \wedge c \bullet s^{\sqsubseteq} \subseteq P \\
\Leftrightarrow & \quad \{\!\![\ c^{\sqsubseteq} = \varepsilon \cup c\ ]\!\!\} \\
 & \varepsilon \in P \wedge c \in P \wedge c \bullet s^{\sqsubseteq} \subseteq P\ .
\end{array}
$$

We assume now that P itself is already given in the form of an automaton-like recursion. Then there is a systematic way for passing from that to a recursion for $\mathsf{saf}\,P$. Suppose that P satisfies, for all $c \in A$ and $U \subseteq A^*$,

$$c \bullet U \subseteq P \Leftrightarrow U \subseteq F_c(P) \tag{8}$$

for some function $F : A \to (\mathcal{P}(A^*) \to \mathcal{P}(A^*))$. In other words, we assume that the "recursive call" $F_c(P)$ depends only on the first symbol of the word to be analyzed. Note that this assumption means a Galois connection between $c\bullet$ and F_c.

Under this assumption we can continue as follows:

$$
\begin{array}{ll}
 & c \bullet s^{\sqsubseteq} \subseteq P \\
\Leftrightarrow & \quad \{\!\![\ \text{by assumption (8)}\ ]\!\!\} \\
 & s^{\sqsubseteq} \subseteq F_c(P) \\
\Leftrightarrow & \quad \{\!\![\ \text{definition}\ ]\!\!\} \\
 & s \in \mathsf{saf}\,F_c(P)\ .
\end{array}
$$

Note that a bi-implication linear in s results. To sum up, we have shown

Lemma 21.3 *Suppose property $P \in \mathcal{P}(A^*)$ satisfies*

$$c \bullet U \subseteq P \Leftrightarrow U \subseteq F_c(P) \ .$$

Then, for $U \neq \emptyset$,

$$\begin{aligned} \varepsilon \in \mathsf{saf}\, P &\Leftrightarrow \varepsilon \in P \ , \\ c \bullet U \subseteq \mathsf{saf}\, P &\Leftrightarrow \varepsilon \in P \wedge c \in P \wedge U \subseteq \mathsf{saf}\, F_c(P) \ . \end{aligned}$$

Assume now that we are given two properties P and Q and seek a recursion for $\mathsf{saf}\, P \cap \mathsf{saf}\, Q = \mathsf{saf}\,(P \cap Q)$. The following result is immediate from Lemma 21.1.5 and Lemma 21.3:

Lemma 21.4 *Suppose P, Q satisfy*

$$(c \bullet U \subseteq P \Leftrightarrow U \subseteq F_c(P)) \wedge (c \bullet U \subseteq Q \Leftrightarrow U \subseteq G_c(P)) \ .$$

Then, for $U \neq \emptyset$,

$$\begin{aligned} \varepsilon \in \mathsf{saf}\,(P \cap Q) &\Leftrightarrow \varepsilon \in P \cap Q \ , \\ c \bullet U \subseteq \mathsf{saf}\,(P \cap Q) &\Leftrightarrow c^{\leq} \subseteq P \cap Q \wedge U \subseteq \mathsf{saf}\,(F_c(P) \cap G_c(Q)) \ . \end{aligned}$$

This corresponds to the construction of a product automaton.

22 Liveness

22.1 Definition and Topological Properties

Following again [1] we call a behaviour $\mathcal{B}$ over streams *live* iff

$$\forall\, s \in A^* : \exists\, T \in A^\omega : s \circ T \in \mathcal{B} \ .$$

Using again (7) we can reduce this to

$$\forall\, s \in A^* : s \in \mathsf{pref}\, \mathcal{B}$$

and hence to

$$A^* \subseteq \mathsf{pref}\, \mathcal{B} \ .$$

Since A^* is the set of compact elements of A^∞ we can again easily generalise this to arbitrary ideal completions. Consider a partial order $(M, \leq)$ and a behaviour $\mathcal{B} \subseteq M^\infty$. Then $\mathcal{B}$ is called *live* iff

$$M \subseteq \mathsf{pref}\, \mathcal{B} \ .$$

We show now (see again [1])

Lemma 22.1 *$\mathcal{B}$ is live iff it is topologically dense in M^∞, ie. iff $\mathsf{should}\, \mathcal{B} = M^\infty$.*

Proof. $M \subseteq \mathsf{pref}\,\mathcal{B}$

$\Leftrightarrow$ $\{\![$ for $(\Rightarrow)$ use transitivity of inclusion,
for $(\Leftarrow)$ the principal ideals $J = x^{\leq}$ for $x \in \mathsf{pref}\,\mathcal{B}$ $]\!\}$

$\forall\ J \in M^{\infty} : J \subseteq \mathsf{pref}\,\mathcal{B}$

$\Leftrightarrow$ $\{\![$ by Corollary 20.3 and the definition of should $]\!\}$

$\forall\ J \in M^{\infty} : J \in \mathsf{should}\,\mathcal{B}$

$\Leftrightarrow$ $\{\![$ set theory $]\!\}$

$M^{\infty} \subseteq \mathsf{should}\,\mathcal{B}$

$\Leftrightarrow$ $\{\![$ set theory $]\!\}$

$M^{\infty} = \mathsf{should}\,\mathcal{B}$.

$\square$

Now we obtain

Lemma 22.2 *Every behaviour is the intersection of a live and a safe behaviour.*

Proof. We could copy the proof of the respective theorem in [1] verbatim, since it proceeds purely in topological terms. However, we give a simpler proof that avoids most of the topological reasoning in [1].
Assume $\mathcal{B} \subseteq M^{\infty}$. We have

$\mathcal{B}$

$=$ $\{\![$ since $\mathcal{B} \subseteq \mathsf{should}\,\mathcal{B}$ $]\!\}$

$\mathsf{should}\,\mathcal{B} \backslash (\mathsf{should}\,\mathcal{B} \backslash \mathcal{B})$

$=$ $\{\![$ definition of $\backslash$, where $\overline{\mathcal{C}}$ denotes the complement
of $\mathcal{C}$ w.r.t. M^{∞} $]\!\}$

$\mathsf{should}\,\mathcal{B} \cap \overline{\mathsf{should}\,\mathcal{B} \cap \overline{\mathcal{B}}}$

$=$ $\{\![$ de Morgan and double complement $]\!\}$

$\mathsf{should}\,\mathcal{B} \cap (\overline{\mathsf{should}\,\mathcal{B}} \cup \mathcal{B})$.

Since $\mathsf{should}\,\mathcal{B}$ is safe, the claim is shown if $\overline{\mathsf{should}\,\mathcal{B}} \cup \mathcal{B}$ is live. We calculate

$\mathsf{should}\,(\overline{\mathsf{should}\,\mathcal{B}} \cup \mathcal{B})$

$\supseteq$ $\{\![$ since $\subseteq$-monotonic and hence superdistributive over $\cup$ $]\!\}$

$\mathsf{should}\,(\overline{\mathsf{should}\,\mathcal{B}}) \cup \mathsf{should}\,\mathcal{B}$

$\supseteq$ $\{\![$ since should is extensive $]\!\}$

$\overline{\mathsf{should}\,\mathcal{B}} \cup \mathsf{should}\,\mathcal{B}$

$=$ $\{\![$ definition of complement $]\!\}$

M^{∞} ,

so that we are done by Lemma 22.1. $\square$

Inspection of the proof leads to the following abstraction. Consider a Boolean algebra $(K, \leq)$ with greatest element $\top$. Call a function $f : K \to K$ a *pre-closure* if it is extensive, ie. satisfies $\forall\, x : x \leq f(x)$, and monotonic. Next, say that $y \in K$ is *f-dense* if $f(y) = \top$. Then we have

Corollary 22.3 *Every element x of K is the meet of an f-image and an f-dense element, viz.*

$$x \;=\; f(x) \sqcap (\overline{f(x)} \sqcup x)\ .$$

Another way of replacing the topological proof of Lemma 22.2 in [1] by a proof over Boolean algebras is presented in [24]. However, our proof is simpler still.

22.2 Liveness and Snapshot Sets

As in the case of safety, we now investigate when a property P spans a live behaviour. We calculate

$\mathsf{ide}\, P$ is live

$\Leftrightarrow$ $\{\!\![$ definition $]\!\!\}$

$M \subseteq \mathsf{pref}\,\mathsf{ide}\, P$

$\Leftrightarrow$ $\{\!\![$ by Lemma 11.4.2 $]\!\!\}$

$M \subseteq P^{\leq}$

$\Leftrightarrow$ $\{\!\![$ definition of $\leq$ $]\!\!\}$

$M \leq P$.

Hence we call $P \subseteq M$ a *liveness property* iff $M \leq P$.

22.3 Spanning Infinite Behaviours by Snapshot Sets

We now define that part of a snapshot set that is relevant for the infinite streams. We call a set $Q \subseteq M$ *lively* iff $Q \neq \emptyset \wedge \mathsf{max}\, Q = \emptyset$.

Lemma 22.4 *1. If Q is lively and $x \in Q$ then there is an $I \in \mathsf{inf}\,\mathsf{ide}\, Q$ with $x \in I$.*

2. If Q is lively then $\mathsf{inf}\,\mathsf{ide}\, Q \neq \emptyset$.

3. If M itself is lively then for every $\mathcal{B}$ we have $M^\infty \subseteq \mathcal{B}$ iff $\mathsf{inf}\, M^\infty \subseteq \mathsf{inf}\, \mathcal{B}$.

Proof. 1. We construct a chain $(x_i)_{i \in \mathbb{N}}$ as follows. Choose $x_0 \overset{\text{def}}{=} x$. Assume now that x_i has been chosen. Since $x_i \notin \mathsf{max}\, Q = \emptyset$, there is an $x_{i+1} \in Q$ with $x_i < x_{i+1}$.

By construction then $K \overset{\text{def}}{=} \{x_i : i \in \mathbb{N}\} \in \mathsf{dir}\, Q$ and hence $I \overset{\text{def}}{=} K^{\leq} \in \mathsf{ide}\, Q$. Moreover, $\mathsf{max}\, I = \mathsf{max}\, K = \emptyset$, i.e, $I \in \mathsf{inf}\,\mathsf{ide}\, Q$.

2. Immediate from 1.
3. Immediate from 1.

□

In connection with the results below, property 3. will allow easier liveness proofs. Note that this is particularly relevant for the case of streams, since the set A^* of compact elements itself is lively.

Now we define the *live part* of $P \subseteq M$ as

$$\mathsf{liv}\, P \stackrel{\text{def}}{=} \bigcup \mathcal{L}_P$$

where

$$\mathcal{L}_P \stackrel{\text{def}}{=} \{Q \subseteq P : Q \text{ lively}\}\ .$$

This operation enjoys the following properties:

Lemma 22.5 *1.* $\mathsf{liv}\, P \subseteq P$.
2. $\mathsf{max}\,\mathsf{liv}\, P = \emptyset$.
3. P *is lively iff* $P \neq \emptyset \wedge P = \mathsf{liv}\, P$.
4. liv *is* $\subseteq$*-monotonic.*
5. $\mathsf{liv}\,\mathsf{liv}\, P = \mathsf{liv}\, P$.
6. $\mathsf{liv}\, P \neq \emptyset \Rightarrow \mathsf{inf}\,\mathsf{ide}\, P \neq \emptyset$.
7. $\mathcal{L}_P \neq \emptyset \Rightarrow \mathsf{inf}\,\mathsf{ide}\, P \neq \emptyset$.
8. $\mathsf{inf}\,\mathsf{ide}\, P = \mathsf{inf}\,\mathsf{ide}\,\mathsf{liv}\, P$.
9. $\mathsf{pref}\,\mathsf{inf}\,\mathsf{ide}\, P = (\mathsf{liv}\, P)^{\leq}$.

Proof. 1. Clear from the definition.
2. Assume $x \in \mathsf{max}\,\mathsf{liv}\, P \subseteq \mathsf{liv}\, P$. Then there is a $Q \in \mathcal{L}_P$ with $x \in Q$. Since $x \notin \emptyset = \mathsf{max}\, Q$, there is $y \in Q \subseteq P$ with $x < y$. Contradiction!
3. The implication $(\Rightarrow)$ is clear.
For the converse we use that $\mathsf{max}\, P = \mathsf{max}\,\mathsf{liv}\, P = \emptyset$ by 2.
4. We have $P \subseteq Q \Rightarrow \mathcal{L}_P \subseteq \mathcal{L}_Q \Rightarrow \bigcup \mathcal{L}_P \subseteq \bigcup \mathcal{L}_Q$.
5. By 1 and 4 we have $\mathsf{liv}\,\mathsf{liv}\, P \subseteq \mathsf{liv}\, P$. For the converse we calculate

$$\begin{aligned}
& Q \subseteq P \wedge Q \text{ lively} \\
\Rightarrow \quad & \{\![\ \text{definition of } \mathsf{liv}\, P\ ]\!\} \\
& Q \subseteq \mathsf{liv}\, P \wedge Q \text{ lively} \\
\Rightarrow \quad & \{\![\ \text{definition of } \mathcal{L}\ ]\!\} \\
& \mathcal{L}_P \subseteq \mathcal{L}_{\mathsf{liv}\, P} \\
\Rightarrow \quad & \{\![\ \text{monotonicity of } \textstyle\bigcup \text{ and definition of } \mathsf{liv}\ ]\!\} \\
& \mathsf{liv}\, P \subseteq \mathsf{liv}\,\mathsf{liv}\, P\ .
\end{aligned}$$

6. By 6 we have $\mathsf{max}\,\mathsf{liv}\, P = \emptyset$. Now from Lemma 22.4 and using $\subseteq$-monotonicity of the $\mathsf{inf}\,\mathsf{ide}$-operation we get $\emptyset \neq \mathsf{inf}\,\mathsf{ide}\,\mathsf{liv}\, P \subseteq \mathsf{inf}\,\mathsf{ide}\, P$.
7. First note that $\emptyset \notin \mathcal{L}_P$. But then $\bigcup \mathcal{L}_P \neq \emptyset$ iff $\mathcal{L}_P \neq \emptyset$. Now apply 6.

8. By 1 and $\subseteq$-monotonicity of $\mathsf{inf}\,\mathsf{ide}$ we get $\supseteq$. Assume, conversely, $I \in \mathsf{inf}\,\mathsf{ide}\,P$. Then there is a $D \in \mathsf{dir}\,P$ with $I = D^{\leq}$. Since D is directed, we have $D \neq \emptyset$. Moreover, $\mathsf{max}\,D = \mathsf{max}\,I = \emptyset$. So $D \in \mathcal{L}_P$ and hence also $D \subseteq \mathsf{liv}\,P$, ie. $D \in \mathsf{dir}\,\mathsf{liv}\,P$. So $I \in \mathsf{inf}\,\mathsf{str}\,\mathsf{liv}\,P$ as well.
9. $$\begin{array}{cl} & \mathsf{pref}\,\mathsf{inf}\,\mathsf{ide}\,P \\ = & \quad \{\!\![\text{ definitions }]\!\!\} \\ & \bigcup\{D^{\leq} : D \in \mathsf{dir}\,P \wedge \mathsf{max}\,D = \emptyset\} \\ = & \quad \{\!\![\text{ distributivity }]\!\!\} \\ & (\bigcup\{D : D \in \mathsf{dir}\,P \wedge \mathsf{max}\,D = \emptyset\})^{\leq} \\ \subseteq & \quad \{\!\![\text{ definition of } \mathcal{L}]\!\!\} \\ & (\bigcup \mathcal{L}_P)^{\leq} \\ = & \quad \{\!\![\text{ definition of } \mathsf{liv}]\!\!\} \\ & (\mathsf{liv}\,P)^{\leq}\ . \end{array}$$
Assume conversely $y \in \mathsf{liv}\,P^{\leq}$. There is a $Q \in \mathcal{L}_P$ and an $x \in Q$ with $y \leq x$. By 22.4.1 there is an $I \in \mathsf{inf}\,\mathsf{ide}\,Q \subseteq \mathsf{inf}\,\mathsf{ide}\,P$ with $x \in I$.

□

So in particular, liv is again a kernel operator. Moreover, by 7, to show that a snapshot set P spans infinite ideals, it suffices to exhibit a lively $Q \subseteq P$. Such Qs can frequently be constructed by induction.

Part V: Extended Example: Buffers and Queues

23 Specification of a Bounded Buffer

As an example of the use of our constructs, we give a specification of bounded buffer and queue modules. For this we use the particular domain $A^{\infty} = A^* \cup A^{\omega}$ of finite and infinite streams over a set A of atomic actions. This example uses the trace view (cf. Section 6) of streams. It was motivated by the asynchronous bounded queue implementation in the collection of the IFIP WG10.5 benchmark problems in hardware verification [28].

The buffer module has one input and one output port. In describing such modules, we choose the letters a for the action of inputting and b for outputting and set $A \overset{\text{def}}{=} \{a, b\}$. Boundedness of a module can be enforced by requiring the number of input actions to exceed the number of output actions by at most some $n \in \mathbb{N}$ which then is the *capacity* of the device.

We denote by s_c the number of occurrences of $c \in A$ in $s \in A^*$. Formally,

$$\begin{aligned} \varepsilon_c &\overset{\text{def}}{=} 0\ , \\ (a \bullet s)_c &\overset{\text{def}}{=} \delta_{ac} + s_c\ , \end{aligned}$$

where δ is the Kronecker symbol defined by

$$\delta_{xy} \stackrel{\text{def}}{=} \begin{cases} 1 \text{ if } x = y \ , \\ 0 \text{ otherwise} \ . \end{cases}$$

Generalising the above informal description slightly, we define, for $n \in \mathbb{Z}$ and $a, b \in A$, the set

$$\mathrm{EX}_n^{ab} \stackrel{\text{def}}{=} \{s \in A^* : s_a \le s_b + n\}$$

of snapshots. Then $s \in \mathrm{EX}_n^{ab}$ may be pronounced as "a exceeds b by at most n in s". The specification is, however, very loose in that the balance between as and bs might be struck only at the very end of a word. For instance, $a^{k+n} \bullet b^k \in \mathrm{EX}_{ab}^n$. So the restriction may be violated in prefixes and only established in the end. For bounded devices, this is not possible. They need a stronger specification. Therefore we strengthen our snapshot set to the safety property

$$\mathrm{B}_n^{ab} \stackrel{\text{def}}{=} \mathsf{saf}\, \mathrm{EX}_n^{ab} \ .$$

Now $\mathsf{ide}\, \mathrm{B}_n^{ab}$ is the set of all finite and infinite streams that satisfy EX_n^{ab} in all prefixes. However, we are interested in devices that work for an unbounded time. This is specified by considering as overall behaviour of such a device the set

$$\mathcal{B}_n^{ab} \stackrel{\text{def}}{=} \mathsf{inf}\, \mathsf{str}\, \mathrm{B}_n^{ab}$$

consisting only of infinite admissible streams.

A buffer is a device in which the number of outputs must not exceed the number of inputs. Hence we define

$$\mathcal{BF}^{ab} \stackrel{\text{def}}{=} \mathcal{B}_0^{ba} \ .$$

Note the reversal of the arguments in the superscript. The finitary property B_0^{ba} spells out to $s_b \le s_a$, as required. This describes an unbounded buffer. A bounded buffer of capacity n then is described by

$$\mathcal{BB}_n^{ab} \stackrel{\text{def}}{=} \mathcal{BF}^{ab} \cap \mathcal{B}_n^{ab} \ .$$

This specifies the set of all infinite streams for which, in all finite prefixes, the number of outputs does not exceed the number of inputs and the number of inputs does not exceed the number of outputs by more than n.

24 Transformation to Automaton Form

We consider now again the family of properties EX_n^{ab}. From its predicative definition in the previous section we want to calculate a more "operational" description corresponding to a generating grammar or accepting automaton. This can be done by a simple unfold/fold transformation using induction on the words in A^*. For the induction basis we calculate

$$\begin{array}{ll}
& \varepsilon \in \mathrm{EX}_n^{ab} \\
\Leftrightarrow & \quad \{\!\![\text{ definition of EX }]\!\!\} \\
& \varepsilon_a \le \varepsilon_b + n \\
\Leftrightarrow & \quad \{\!\![\text{ definition of count }]\!\!\} \\
& 0 \le 0 + n \\
\Leftrightarrow & \quad \{\!\![\text{ arithmetic }]\!\!\} \\
& 0 \le n \ .
\end{array}$$

For the induction step, we consider an arbitrary $c \in A$:

$$\begin{array}{ll}
& c \bullet s \in \mathrm{EX}_n^{ab} \\
\Leftrightarrow & \quad \{\!\![\text{ definition of EX }]\!\!\} \\
& (c \bullet s)_a \le (c \bullet s)_b + n \\
\Leftrightarrow & \quad \{\!\![\text{ definition of count }]\!\!\} \\
& \delta_{ca} + s_a \le \delta_{cb} + s_b + n \\
\Leftrightarrow & \quad \{\!\![\text{ arithmetic }]\!\!\} \\
& s_a \le s_b + n + \delta_{cb} - \delta_{ca} \\
\Leftrightarrow & \quad \{\!\![\text{ definition of EX }]\!\!\} \\
& s \in \mathrm{EX}_{n+\delta_{cb}-\delta_{ca}}^{ab} \ ,
\end{array}$$

Note that the recursion relations are linear bi-implications. Therefore we obtain, for $U \neq \emptyset$,

$$\begin{array}{c}
\varepsilon \in \mathrm{EX}_n^{ab} \Leftrightarrow 0 \le n \ , \\
c \bullet U \subseteq \mathrm{EX}_n^{ab} \Leftrightarrow U \subseteq \mathrm{EX}_{n+\delta_{cb}-\delta_{ca}}^{ab} \ .
\end{array}$$

This corresponds to an infinite grammar with nonterminals EX_n^{ab} or an infinite automaton with states EX_n^{ab} ($n \in \mathbb{Z}$).

25 Counting Resumed

Next we want a similar representation for

$$\mathrm{B}_n^{ab} \overset{\mathrm{def}}{=} \mathsf{saf}\,\mathrm{EX}_n^{ab} \ .$$

This can be done quite systematically using Lemma 21.3. We obtain, for $U \neq \emptyset$,

$$\begin{array}{rl}
\varepsilon \in \mathrm{B}_n^{ab} & \Leftrightarrow 0 \le n \ , \\
c \bullet U \subseteq \mathrm{B}_n^{ab} & \Leftrightarrow 0 \le n \wedge 0 \le n \wedge U \subseteq \mathrm{B}_n^{ab} \text{ for } c \in A \backslash \{a, b\} \ , \\
a \bullet U \subseteq \mathrm{B}_n^{ab} & \Leftrightarrow 0 \le n \wedge 0 \le n-1 \wedge U \subseteq \mathrm{B}_{n-1}^{ab} \ , \\
b \bullet U \subseteq \mathrm{B}_n^{ab} & \Leftrightarrow 0 \le n \wedge 0 \le n+1 \wedge U \subseteq \mathrm{B}_{n+1}^{ab} \ .
\end{array}$$

which simplifies to

$$\begin{array}{rl} \varepsilon \in \mathrm{B}^{ab}_n \Leftrightarrow & 0 \leq n \ , \\ c \bullet U \subseteq \mathrm{B}^{ab}_n \Leftrightarrow & 0 \leq n \wedge U \subseteq \mathrm{B}^{ab}_n \text{ for } c \in A \backslash \{a, b\} \ , \\ a \bullet U \subseteq \mathrm{B}^{ab}_n \Leftrightarrow & 0 \leq n-1 \wedge U \subseteq \mathrm{B}^{ab}_{n-1} \ , \\ b \bullet U \subseteq \mathrm{B}^{ab}_n \Leftrightarrow & 0 \leq n \wedge U \subseteq \mathrm{B}^{ab}_{n+1} \ . \end{array}$$

In particular, $\mathrm{B}^{ab}_n = \emptyset$ for $n < 0$.

Now we consider the bounded buffer behaviour. We calculate:

$$\begin{array}{ll} & \mathcal{BB}^{ab}_n \\ = & \quad \{\![\text{ definition }]\!\} \\ & \mathcal{BF}^{ab} \cap \mathcal{B}^{ab}_n \\ = & \quad \{\![\text{ definition }]\!\} \\ & \mathcal{B}^{ba}_0 \cap \mathcal{B}^{ab}_n \\ = & \quad \{\![\text{ definition }]\!\} \\ & \mathsf{inf\,str}\,\mathrm{B}^{ba}_0 \cap \mathsf{inf\,str}\,\mathrm{B}^{ab}_n \\ = & \quad \{\![\text{ by Lemma 12.3.5, since the B sets have been specified as safety} \\ & \quad\;\; \text{properties }]\!\} \\ & \mathsf{inf\,str}\,(\mathrm{B}^{ba}_0 \cap \mathrm{B}^{ab}_n) \ . \end{array}$$

So the problem has been reduced to finding an explicit representation for $\mathrm{B}^{ba}_0 \cap \mathrm{B}^{ab}_n$, which is a simple product automaton construction. It is a special case of the automaton for

$$\mathrm{G}_{mn} \stackrel{\text{def}}{=} \mathrm{B}^{ba}_m \cap \mathrm{B}^{ab}_n \ .$$

26 Decomposition

Let us now define a buffer process by setting

$$\mathrm{BB}^{ab}_m \stackrel{\text{def}}{=} (\{a, b\}, \mathcal{BB}^{ab}_m) \ .$$

Then using parallel composition we can state the following nice decomposition properties:

Lemma 26.1 *1.* $\mathrm{EX}^{ab}_m \cap \mathrm{EX}^{bc}_n \subseteq \mathrm{EX}^{ac}_{m+n}$.
2. $\mathrm{B}^{ab}_m \cap \mathrm{B}^{bc}_n \subseteq \mathrm{B}^{ac}_{m+n}$.
3. $S \dagger \{a, b\} \in \mathcal{BB}^{ab}_m \wedge S \dagger \{b, c\} \in \mathcal{BB}^{bc}_n \Rightarrow S \dagger \{a, c\} \in \mathcal{BB}^{ac}_{m+n}$.
4. $\mathrm{BB}^{ab}_m \mid\mid \mathrm{BB}^{bc}_n \subseteq \mathrm{BB}^{ac}_{m+n}$.

Proof. 1. $\quad s \in \mathrm{EX}^{ab}_m \cap \mathrm{EX}^{bc}_n$

$\Leftrightarrow \quad \{\![\text{ definition }]\!\}$

$$
\begin{array}{ll}
& s_a \le s_b + m \wedge s_b \le s_c + n \\
\Rightarrow & \quad \{\!\![\text{ transitivity and monotonicity }]\!\!\} \\
& s_a \le s_c + m + n \\
\Leftrightarrow & \quad \{\!\![\text{ definition }]\!\!\} \\
& s \in \mathrm{EX}^{ac}_{m+n} \ .
\end{array}
$$

2. immediate from 1.,Lemma 21.1.5 and Lemma 21.1.4.
3. immediate from 2.
4. immediate from 3.

□

This allows decomposing a buffer of capacity n into a parallel composition of n buffers of capacity 1. Of course, it needs to be shown that the intersections/parallel compositions are non-empty. This follows from our results in Section 22: first, it is easy to show that

$$
\begin{array}{rcl}
(a \bullet b)^* \subseteq \mathrm{EX}^{ab}_n & \Leftarrow & n \ge 0 \ , \\
\mathsf{inf\,str}\,(a \bullet b)^* \subseteq \mathcal{BB}^{ab}_n & \Leftarrow & n \ge 1 \ .
\end{array}
$$

From this we get

$$
\mathsf{inf\,str}\,(a \bullet b \bullet c)^* \subseteq \beta(\mathrm{BB}^{ab}_m \mid\mid \mathrm{BB}^{bc}_n) \quad \Leftarrow \quad m, n \ge 1 \ .
$$

Since $(a \bullet b \bullet c)^*$ is lively, $\mathsf{inf\,str}\,(a \bullet b \bullet c)^*$ and hence $\mathrm{BB}^{ab}_m \mid\mid \mathrm{BB}^{bc}_n$ are non-empty.

27 The One-Place Buffer

For the special case of $n = 1$ we have

$$
\mathcal{BB}^{ab}_1 = \mathsf{inf\,str}\,\mathrm{G}_{01} \ ,
$$

where, for $U \ne \emptyset$,

$$
\begin{array}{rcl}
\varepsilon \in \mathrm{G}_{01} \Leftrightarrow \mathrm{TRUE} \ , & \qquad & \varepsilon \in \mathrm{G}_{10} \Leftrightarrow \mathrm{TRUE} \ , \\
a \bullet U \subseteq \mathrm{G}_{01} \Leftrightarrow U \subseteq \mathrm{G}_{10} \ , & & a \bullet U \subseteq \mathrm{G}_{10} \Leftrightarrow \mathrm{FALSE} \ , \\
b \bullet U \subseteq \mathrm{G}_{01} \Leftrightarrow \mathrm{FALSE} \ , & & b \bullet U \subseteq \mathrm{G}_{10} \Leftrightarrow U \subseteq \mathrm{G}_{01} \ .
\end{array}
$$

This corresponds to a two-state accepting automaton for the bounded buffer property, which is sufficient for purposes of implementation.

However, the above can also be seen as a regular grammar or system of equations for languages. We can calculate from it a regular expression for G_{01} using twice

$$
\textbf{(Arden's Rule)} \quad \frac{\varepsilon \ne U \qquad X = V \cup U \bullet X}{X = U^* \bullet V \ .}
$$

This gives

$$
\mathrm{G}_{01} = (a \bullet b)^* \bullet (\varepsilon \cup a) \ .
$$

Using Example 14.5 and Corollary 14.4, we obtain

$$\mathcal{BB}_1^{ab} \;=\; \mathsf{inf}\,\mathsf{str}\,(a \bullet b)^* \ .$$

Finally we use the fact that the language $a \bullet b$ as a singleton trivially satisfies the Fano condition, so that Lemma 16.1 gives

$$\mathcal{BB}_1^{ab} \;=\; (a \bullet b)^\omega \ ,$$

as expected.

28 From Buffers to Queues

So far we have only talked about the relative order of input and output *events*. For queues also the relative order of input and output *values* is relevant. We use now the refined alphabet $A = C \times V$ where C is the set of channel names and V the set of values. An element of A will be denoted as $c\langle v\rangle$. As a shorthand we introduce

$$c \;=\; \{c\langle x\rangle : x \in V\} \ . \tag{9}$$

For a word $w \in A^*$ we define the word $\mathit{chans}(w)$ of channels on which activity occurred and for each $c \in C$ the word $\mathit{vals}_c(W)$ of values transmitted along c. Their inductive definitions read

$$\begin{aligned} \mathit{chans}(\varepsilon) &= \varepsilon \ , \\ \mathit{chans}(b\langle x\rangle \bullet w) &= b \bullet \mathit{chans}(w) \ , \end{aligned}$$

$$\begin{aligned} \mathit{vals}_c(\varepsilon) &= \varepsilon \ , \\ \mathit{vals}_c(b\langle x\rangle \bullet w) &= \begin{cases} x \bullet \mathit{vals}_c(w) & \text{if } b = c \ , \\ \mathit{vals}_c(w) & \text{otherwise} \ . \end{cases} \end{aligned}$$

These operations are extended pointwise to languages and behaviours.

With these operations we may specify the behaviour of a *faithful* component, ie. a component which does not re-order or lose messages when transmitting from channel a to channel b, as

$$\mathcal{FA}^{ab} \overset{\text{def}}{=} \{S : \mathit{vals}_a(S) \;=\; \mathit{vals}_b(S)\} \ .$$

A bounded queue is then specified as a faithful bounded buffer:

$$\mathcal{BQ}_n^{ab} \overset{\text{def}}{=} \mathcal{FA}^{ab} \cap \mathcal{BB}_n^{ab} \ .$$

Here a, b in $\mathcal{BB}_n^{ab}$ are to be understood according to abbreviation (9).

The decomposition properties for buffers carry over to queues, so that again a queue of capacity n can be refined into the parallel composition of n queues of capacity 1. Moreover, a similar calculation as before, using again Arden's rule, yields for the refinement

$$\mathcal{BQ}_1^{ab} \;=\; (\bigcup_{x \in V} a\langle x\rangle \bullet b\langle x\rangle)^\omega \ .$$

29 Conclusion

We have introduced some algebraic operators and laws that can be used in the specification and derivation of systems. By abstracting from the domain of streams for which most of the notions were coined originally, we have obtained a rich set of laws which hold for a variety of domains. The order-theoretic approach lends itself well to an algebraic treatment. The point-free formulation eases and compacts specifications, proofs of the basic properties and the actual derivations. Further research along these lines should search for similar algebraic characterisations of other important notions about systems and to explore their algebraic properties.

Concerning the underlying theory, our domain-theoretic notions should be tied in more closely with the topological view (see eg. [55, 59]). Moreover, in the stream domain there obviously is a close connection with temporal operators: $\mathsf{str}\, P$ is related to intermittent assertions [15] and hence the formula $\Box\Diamond P$ (always eventually P) in temporal logic, while $\mathsf{saf}\, P$ corresponds to $\boxed{\mathrm{i}}\, P$ (P holds in all initial subintervals [48]). These connections are made precise and carried over to arbitrary domains in [44, 45]. The resulting "modal algebra" as well as the ideal and stream algebra developed in the present paper need to be tried out in larger and more realistic case studies of deductive design of parallel systems.

Acknowledgement: Many helpful remarks on this paper were provided by J. Baeten, A. Ponse and V. Stoltenberg-Hansen.

References

1. B. Alpern, F.B. Schneider: Defining liveness. Information Processing Letters **21**, 181–185 (1985)
2. R.-J.R. Back: A calculus of refinements for program derivations. Acta Informatica **25**, 593–624 (1988)
3. J.C.M. Baeten, W.P. Wijland: Process algebra. Cambridge Tracts in Theoretical Computer Science **18**. Cambridge: Cambridge University Press 1990
4. J.W. de Bakker, J.I. Zucker: Compactness in semantics for merge and fair merge. In: E. Clarke and D. Kozen (eds.): Logics of Programs. Lecture Notes in Computer Science **164**. Berlin: Springer 1983, 18-33
5. F.L. Bauer, B. Möller, H. Partsch, P. Pepper: Formal program construction by transformations — Computer-aided, Intuition-guided Programming. IEEE Transactions on Software Engineering **15**, 165–180 (1989)
6. G. Birkhoff: Lattice theory, 3rd edition. American Mathematical Society Colloquium Publications, Vol. XXV. Providence, R.I.: AMS 1967
7. R. Bird: Lectures on constructive functional programming. In: M. Broy (ed.): Constructive methods in computing science. NATO ASI Series. Series F: Computer and Systems Sciences **55**. Berlin: Springer 1989, 151–216
8. R.S. Bird, O. de Moor: Algebra of programming. Prentice-Hall 1996
9. J.D. Brock, W.B. Ackerman: Scenarios: a model of non-determinate computation. In: J. Díaz, I. Ramos (ed.): Formalization of programming concepts. Lecture Notes in Computer Science **107**. Berlin: Springer 1981, 252–259

10. M. Broy: Specification and refinement of a buffer of length one. In: M. Broy (ed.): Deductive program design. NATO ASI Series, Series F: Computer and Systems Sciences **152**. Berlin: Springer 1996, 273–304
11. M. Broy: Functional specification of time sensitive communicating systems. In: M. Broy (ed.): Programming and mathematical method. NATO ASI Series, Series F: Computer and Systems Sciences **88**. Berlin: Springer 1992, 325–367
12. M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T.F. Gritzner, R. Weber: The design of distributed systems — an introduction to FOCUS. Revised Version. Institut für Informatik der TU München, Report TUM-I9202-2 (SFB-Bericht Nr. 342/2-2/92 A), 1993
13. M. Broy, G. Ştefănescu: The algebra of stream processing functions. Institut für Informatik, TU München, Report TUM-I9620, 1996
14. M. Broy, K. Stølen: Specification and refinement of finite dataflow networks — a relational approach. In: H. Langmaack, W.-P. de Roever, J. Vytopil (eds.): Formal techniques in real-time and fault-tolerant computing. Lecture Notes in Computer Science **863**. Berlin: Springer 1994, 247–267
15. R.M. Burstall: Program proving as hand simulation with a little induction. Proc. IFIP Congress 1974. Amsterdam: North-Holland1974, 308–312
16. R.M. Burstall, J. Darlington: A transformation system for developing recursive programs. J. ACM **24**, 44–67 (1977)
17. K.M. Chandy, J. Misra: Parallel program design: a foundation. Reading, Mass.: Addison Wesley 1988
18. J.H. Conway: Regular algebra and finite machines. London: Chapman and Hall 1971
19. B.A. Davey, H.A. Priestley: Introduction to lattices and order. Cambridge: Cambridge University Press 1990
20. M. Davis: Infinitary games of perfect information. In: M. Dresher, L.S. Shapley, A.W. Tucker (eds.): Advances in game theory. Princeton, N.J.: Princeton University Press 1964, 89–101
21. F. Dederichs, R. Weber: Safety and liveness from a methodological point of view. Information Processing Letters **36**, 25–30 (1990)
22. E.A. Emerson: Temporal and modal logic. In: J. van Leeuwen (ed.): Handbook of theoretical computer science. Volume B: Formal models and semantics. Amsterdam: Elsevier 1990, 995–1072
23. M.S. Feather: A survey and classification of some program transformation approaches and techniques. In L.G.L.T. Meertens (ed.): Proc. IFIP TC2 Working Conference on Program Specification and Transformation, Bad Tölz, April 14–17, 1986. Amsterdam: North-Holland 1987, 165–195
24. H.P. Gumm: Another glance at the Alpern-Schneider characterization of safety and liveness in concurrent executions. Information Processing Letters **47**, 291–294 (1993)
25. C.A.R. Hoare: Communicating sequential processes. London: Prentice Hall 1985
26. C.A.R. Hoare: Conjunction and concurrency. PARBASE 90, 1990
27. J.K. Huggins: Kermit: specification and verification. In: E. Börger (ed.): Specification and validation methods. Oxford: Clarendon Press 1995
28. IFIP 94/97: IFIP WG 10.5 Verification Benchmarks. Web document under `http://goethe.ira.uka.de/hvg/benchmarks.html`
29. B. Jonsson: A fully abstract trace model for dataflow and asynchronous networks. Distributed Computing **7**, 197–212 (1994)

30. G. Kahn: The semantics of a simple language for parallel processing. In: J.L. Rosenfeld (ed.): Information Processing 74. Proc. IFIP Congress 1974. Amsterdam: North-Holland 1974, 471–475
31. B. Von Karger, C.A.R. Hoare: Sequential calculus. Information Processing Letters **53**, 123–130 (1995)
32. J.N. Kok: A fully abstract semantics for data flow nets. In: J.W. de Bakker, A.J. Nijman, P.C. Treleaven (eds.): PARLE, Parallel languages and architectures Europe, Volume I. Lecture Notes in Computer Science **259**. Berlin: Springer 1987, 351–368
33. L. Lamport: Proving the correctness of multiprocess programs. IEEE Trans. Software Eng. **SE-3**, 125–143 (1977)
34. L. Lamport: Specifying concurrent program modules. ACM TOPLAS **5**, 190–222 (1983)
35. L.G.L.T. Meertens: Algorithmics — Towards programming as a mathematical activity. In: J. W. de Bakker et al. (eds.): Proc. CWI Symposium on Mathematics and Computer Science. CWI Monographs Vol **1**. Amsterdam: North-Holland 1986, 289–334
36. R. Milner: Communication and concurrency. London: Prentice Hall 1989
37. B. Möller: Relations as a program development language. In [38], 373–397
38. B. Möller (ed.): Constructing programs from specifications. Proc. IFIP TC2/WG 2.1 Working Conference on Constructing Programs from Specifications, Pacific Grove, CA, USA, 13–16 May 1991. Amsterdam: North-Holland 1991, 373–397
39. B. Möller: Derivation of graph and pointer algorithms. In: B. Möller, H.A. Partsch, S.A. Schuman (eds.): Formal program development. Lecture Notes in Computer Science **755**. Berlin: Springer 1993, 123–160
40. B. Möller: Algebraic calculation of graph and sorting algorithms. In: D. Bjørner, M. Broy, I.V. Pottosin (eds.): Formal Methods in Programming and their Applications. Lecture Notes in Computer Science **735**. Berlin: Springer 1993, 394–413
41. B. Möller, M. Russling: Shorter paths to graph algorithms. In: R.S. Bird, C.C. Morgan, J.C.P. Woodcock (eds.): Mathematics of Program Construction. Lecture Notes in Computer Science **669**. Berlin: Springer 1993, 250–268. Science of Computer Programming **22**, 157–180 (1994)
42. B. Möller: Ideal streams. In: E.-R. Olderog (ed.): Programming concepts, methods and calculi. IFIP Transactions A-56. Amsterdam: North-Holland 1994, 39–58
43. B. Möller: Refining ideal behaviours. Institut für Mathematik der Universität Augsburg, Report Nr. 345, 1995
44. B. Möller: Temporal operators on partial orders. Proc. 3rd Domain Workshop, Munich, May 29-31, 1997. Ludwig-Maximilian-Universität München (to appear)
45. B. Möller: Modal and Temporal Operators on Partial Orders. In: R. Berghammer (ed.): Programmiersprachen und Grundlagen der Programmierung. Institut für Informatik und Praktische Mathematik, Universität Kiel (to appear). Extended version: Institut für Informatik der Universität Augsburg, Report 97-02, 1997
46. C.C. Morgan: Programming from Specifications. Prentice-Hall, 1990.
47. J.M. Morris: A theoretical basis for stepwise refinement and the programming calculus. Science of Computer Programming **9**, 287–306 (1987)
48. B. Moszkowski: Some very compositional temporal properties. In: E.-R. Olderog (ed.): Programming concepts, methods and calculi. IFIP Transactions A-56. Amsterdam: North-Holland 1994, 307–326
49. M. Nivat: Behaviors of processes and synchronized systems of processes. In: M. Broy, G. Schmidt (eds.): Theoretical foundations of programming methodology. Dordrecht: Reidel 1982, 473–551

50. E.-R. Olderog: Nets, terms and formulas. Cambridge: Cambridge University Press 1991
51. E.-R. Olderog, C.A.R. Hoare: Specification-oriented semantics for communicating processes. Acta Informatica **23**, 9–66 (1986)
52. D. Park: On the semantics of fair parallelism. In D. Bjørner (ed.): Abstract software specifications. Lecture Notes in Computer Science **86**. Berlin: Springer 1980, 504–526
53. H.A. Partsch: Specification and transformation of programs — A formal approach to software development. Berlin: Springer 1990
54. G.D. Plotkin: A powerdomain construction. SIAM J. Computing **5**, 452-487 (1976)
55. R. Redziejowski: Infinite-word languages and continuous mappings. Theoretical Computer Science **43**, 59–79 (1986)
56. F.J. Rietman: A relational calculus for the design of distributed algorithms. Dissertation, University of Utrecht, 1995
57. R. Sharp: Principles of Protocol design. London: Prentice Hall 1994
58. M.B. Smyth: Power domains. J. Computer Syst. Sciences **16**, 23–36 (1978)
59. M.B. Smyth: Topology. In: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (eds.): Handbook of logic in computer science. Vol. 1, Background: Mathematical structures. Oxford: Clarendon Press 1992, 641–761
60. L. Staiger: Research in the theory of ω-languages. J. Inf. Process. Cybern. EIK **23**, 415–439 (1987)
61. L. Staiger: ω-languages. In: G. Rozenberg, A. Salomaa (eds.): Handbook of formal languages. Vol. 3: Beyond words. Berlin: Springer 1997, 339–387
62. R. Stephens: A survey of stream processing. Acta Informatica **34**, 491–541 (1997)
63. K. Stølen, M. Fuchs: An exercise in conditional refinement. In: B. Möller, J.V. Tucker (eds.): Prospects for hardware foundations. Lecture Notes in Computer Science **1546**. Berlin: Springer (this volume)
64. V. Stoltenberg-Hansen, I. Lindström and E.R. Griffor: Mathematical theory of domains. Cambridge Tracts in Theoretical Computer Science, Vol. 22. Cambridge: Cambridge University Press 1994
65. W. Thomas: Automata on infinite objects. In: J. van Leeuwen (ed.): Handbook of theoretical computer science. Vol. B: Formal models and semantics. Amsterdam: Elsevier 1990, 133–191
66. W. Thomas: Languages, automata and logic. In: G. Rozenberg, A. Salomaa (eds.): Handbook of formal languages. Vol. 3: Beyond words. Berlin: Springer 1997, 389–455
67. J. Zwiers: Compositionality, concurrency and partial correctness. Lecture Notes in Computer Science **321**. Berlin: Springer 1989

30 Appendix: Auxiliary Lemmas

30.1 Cones and Maximal Elements

Lemma 30.1 *Consider $N, P \subseteq M$. Then*

1. $(N \cup P)^< = N^< \cup P^< \wedge (N \cup P)^\le = N^\le \cup P^\le$ *(distributivity).*
2. $(N^\le)^< = N^< \wedge (N^\le)^\le = N^\le$.

Lemma 30.2 *Consider $N, P \subseteq M$. Then*

1. $\mathsf{max}\, N = N^{\leq} \backslash N^{<}$.
2. $\mathsf{max}\, N = \mathsf{max}\, N^{\leq}$.
3. $N \subseteq P \Rightarrow N \cap \mathsf{max}\, P \subseteq \mathsf{max}\, N$.
4. $\mathsf{max}\, N \cap P^{<} = \emptyset \Rightarrow \mathsf{max}\,(N \cup P) = \mathsf{max}\, N \cup (\mathsf{max}\, P) \backslash N^{<}$.

Lemma 30.3 *Consider $N, P \subseteq M$. Then*

1. $N \leq P \Leftrightarrow N \leq P^{\leq}$.
2. $L \subseteq N \wedge N \leq P \wedge P \subseteq Q \Rightarrow L \leq Q$.
3. $N \leq P \Leftrightarrow N^{\leq} \subseteq P^{\leq}$.
4. $N \leq P \Rightarrow \mathsf{max}\,(N \cup P) = \mathsf{max}\, P$.

Lemma 30.4 *Consider $N, P \subseteq M$. Then $N \sim P \Leftrightarrow N^{\leq} = P^{\leq}$.*

Proof. Immediate from Lemma 30.3.3. □

30.2 Directed Sets

Lemma 30.5 *Consider $N, P \subseteq M$. Then*

1. $N \cup P \in \mathsf{dir}\, M \wedge N \leq P \Rightarrow P \in \mathsf{dir}\, M$.
2. $N \cup P \in \mathsf{dir}\, M \Rightarrow (N \leq P \vee P \leq N) \wedge (N \in \mathsf{dir}\, M \vee P \in \mathsf{dir}\, M)$.
3. $Q^{\leq} \in \mathsf{dir}\, M \Leftrightarrow Q \in \mathsf{dir}\, M$.
4. $N \leq P \wedge P \in \mathsf{dir}\, M \Rightarrow N \cup P \in \mathsf{dir}\, M$.
5. $\mathsf{dir}\,(N \cup P) = \{K \cup L : (K \in \mathsf{dir}\, N \wedge L \subseteq P \wedge L \leq K)\} \cup$
   $\{K \cup L : (L \in \mathsf{dir}\, P \wedge K \subseteq N \wedge K \leq L)\}$.

Proof.
1. Assume $x, y \in P$. By directedness of $N \cup P$ there is a $z \in N \cup P$ with $x \leq z$ and $y \leq z$. If $z \in P$, we are done. Otherwise, by $N \leq P$ there is a $u \in P$ with $z \leq u$ so that by transitivity also $x \leq u$ and $y \leq u$.
2. For $N = \emptyset$ or $P = \emptyset$ the claim is trivial. So consider $N, P \neq \emptyset$ and suppose $N \not\leq P$. Then there is $x \in N$ with $x \not\leq P$. Assume now $y \in P$. By directedness of $N \cup P$ there is a $z \in N \cup P$ with $x, y \leq z$. Since $x \not\leq P$, it follows that $z \in N \backslash P \subseteq N$. Since y was arbitrary, we have shown $P \leq N$.
   The second disjunct is immediate from the first and 1.
3. Immediate from 1 by setting $N = Q^{\leq}$, $P = Q$ and using $Q^{\leq} \leq Q$.
4. Assume $x, y \in N \cup P$. By $N \leq P$ and $P \leq P$ there are $u, v \in P$ with $x \leq u \wedge y \leq v$. Since P is directed, there is $z \in P$ with $u \leq z$ and $v \leq z$. Hence also $x \leq z$ and $y \leq z$ by transitivity.
5. We show $(\subseteq)$; the reverse inclusion is immediate from 4.
   Consider $Q \in \mathsf{dir}\,(N \cup P)$. We have $Q = K \cup L$ where $K \overset{\text{def}}{=} Q \cap N$ and $L \overset{\text{def}}{=} Q \cap P$. By 2 we know $K \leq L \vee L \leq K$. If $K \leq L$ then $L \in \mathsf{dir}\, P$ by 1. If $L \leq K$ then $K \in \mathsf{dir}\, N$ by 1. This shows the claim.

□