THE $\mu$-CALCULUS AS AN ASSERTION LANGUAGE
FOR FAIRNESS ARGUMENTS

F.A.Stomp, W.P.de Roever, and R.T.Gerth

RUU-CS-84-12
November 1984
THE μ-CALCULUS AS AN ASSERTION LANGUAGE
FOR FAIRNESS ARGUMENTS

F.A.Stomp, W.P.de Roever, and R.T.Gerth

Technical Report RUU-CS-84-12
November 1984

Department of Computer Science
University of Utrecht
P.O.Box 80.012, 3508 TA Utrecht
the Netherlands
THE \( \mu \)-CALCULUS AS AN ASSERTION LANGUAGE
FOR FAIRNESS ARGUMENTS

by

F.A. Stomp \- University of Utrecht. )
W.P. de Roever \- University of Nijmegen/
University of Utrecht.
R.T. Gerth \- University of Utrecht.

-November 1984-

Abstract: Various principles of proof have been proposed to reason about
fairness ([2],[5],[7],[12]). This paper addresses \- for the first time \- the question in what formalism such fairness-arguments can be couched. To
wit: we prove that Park's monotone first-order \( \mu \)-calculus ([6],[12]),
augmented with constants for all recursive ordinals can serve as an
assertion-language for proving fair termination of do-loops. In particu-
lar, the weakest precondition for fair termination of a loop w.r.t. some
postcondition is definable.
The relevance of this result to proving eventualities in Manna and
Pnueli's temporal logic formalism ([8]) is discussed.

*) Presently employed at the University of Nijmegen.
1.4 Next, nearing the focus of this paper, the interaction between fairness and the interleaving model must be examined. How does one deduce properties in the resulting model? The properties of interest always contain eventualities which are enforced by the assumption of fairness. Pure invariances, i.e., properties which are invariant during execution, are not influenced by postulating fairness as extra requirement, and can be derived using more traditional methods.

1.5 The state of art offers the following picture:
To establish that for a concurrent program \( \Psi \) eventually holds, i.e., \( \diamond \Psi \) holds, using the eventuality operator \( \diamond \) from temporal logic, where \( \Psi \) is a state formula, i.e., a direct property of the program state not requiring temporal operators s.a. \( \diamond \) anymore for its expression, the following strategy is taken:

(1) Amongst the concurrent processes an (amongst others state dependent) distinction is made between those processes—in Manna and Pnueli's ([8]) terminology dubbed helpful processes—whose execution brings satisfaction of \( \Psi \) always nearer, and those processes that do not do so, i.e., whose execution possibly does not bring satisfaction of \( \Psi \) any nearer, called steady (or unhelpful) processes.

(2) It must be proved that systematically avoiding execution of any helpful process either leads to an interleaving of steady processes which does not satisfy fairness, i.e., is unfair, since infinitely often a helpful process is enabled but not taken, or, due to some nondeterministic choice in a steady process or the interleaving, does bring satisfaction of \( \Psi \) eventually nearer or even eventually establishes \( \Psi \).

Essential is here that upon closer inspection part (2) above requires application of the same strategy to a syntactically simpler program: just remove the helpful processes from the original program and prove that eventually one of the following holds: \( \Psi \), getting nearer to \( \Psi \) or \( a \), helpful process is enabled.

1.6 A technical formulation of this strategy requires the introduction of well-founded sets, and looks as follows ([8]):

**The Well-founded Liveness Principle—WELL**

Let \( W = (A,S) \) be a well-founded ordered structure. Let \( \phi(a) \) be a parametrized state formula (intuitively expressing how far establishing \( \Psi \) is). Let \( h:A \rightarrow \{1,\ldots,k\} \) be a helpfulness function identifying for each \( a \in A \) the helpful process \( P_{h(a)} \) for states satisfying \( \phi(a) \).

\[
(A) \vdash P \text{ leads from } \phi(a) \text{ to } [\Psi \lor (\exists \beta \leq a \phi(\beta))]
\]

\[
(B) \vdash P_{h(a)} \text{ leads from } \phi(a) \text{ to } [\Psi \lor (\exists \beta < a \phi(\beta))]
\]

\[
(C) \vdash \phi(a) \diamond [\Psi \lor (\exists \beta < a \phi(\beta)) \lor \text{Enabled}(P_{h(a)})]
\]

\[
\vdash (\exists a \phi(a)) \diamond \psi
\]
Here $P$ leads from $\phi$ to $\phi'$ means: $P_i$ leads from $\phi$ to $\phi'$, for $i=1,\ldots,k$.

And $P_i$ leads from $\phi$ to $\phi'$ means that every state transition in $P_i$ establishes $\phi'$ afterwards provided $\phi$ is satisfied before (here $\phi'$ and $\phi'$ are state formulae).

The soundness proof of this rule requires induction over well-founded sets.

Conversely, given the fact that $\Diamond \psi$ is valid, (naive) set theory is used to argue the existence of the required auxiliary quantities (the well-founded ordered structure $\mathbf{W}$, the ranking predicate $p(a)$, and the helpfulness function $h$) which satisfy clauses (A), (B), (C), so that for each such $\psi$ WELL can always be applied. This proves that WELL is semantically complete.

Manna and Pnueli ([8]) even prove that, for certain classes of formulae, their temporal logic formalism is relative complete.

Relative means here: all valid temporal formulae with the given domain interpretation are taken as axioms. Typically, their proof shows that issues concerning programs and executions can be reduced via their rules (from which the one above is derived and deals with eventualities) to state assertions concerning the given program, the so-called state properties.

Now we are ready to ask the one question this paper is about:

How do these results help us if we are sure that $\Diamond \psi$ holds and want to apply the rule above to verify $\Diamond \psi$?

The answer is: not much.

Questions such as:

- How to obtain the appropriate well-founded ordered structure $\mathbf{W}$?
- How does one express, and reason about, the helpfulness-function $h$ and the ranking predicate $p(a)$?
- In general, which assertion language should be used to establish hypotheses (A), (B), (C), of WELL?

are not answered by the above results, since the state properties are not formalized.

The present paper suggests a direction to answer these questions, by concentrating on these problems as they occur when proving termination of do-loops under the above fairness assumptions, i.e., fair termination of do-loops.

That this does not lead to oversimplification follows from the fact that the same auxiliary quantities, with comparable objectives, occur in the rule whose expression and use we shall investigate ([5]):
The Well-founded Liveness Principle for loops—Orna's rule

Let $\mathcal{M} = (A, \preceq)$ be a well-founded structure.
Let $\pi : A \rightarrow (\text{States} \times \{\text{true}, \text{false}\})$ be a predicate, and $q$ be a state predicate.
Let for $w \in A$, with $w$ not minimal (denoted by $0 < w$), be given pairwise disjoint sets $D_w$ and $S_w$, such that $D_w \neq \emptyset$ and $D_w \cup S_w = \{1, \ldots, n\}$.

(a) $\vdash [\pi(w) \wedge w > 0 \wedge b_j] S_j [\exists v < w \pi(v)],$ for all $j \in D_w$

(b) $\vdash [\pi(w) \wedge w > 0 \wedge b_j] S_j [\exists v < w \pi(v)],$ for all $j \in S_w$.

(c) $\vdash [\pi(w) \wedge w > 0] \text{fair}(\bigwedge_{i \in S_w} b_i \wedge \bigwedge_{j \in D_w} \neg b_j \wedge S_j)[\text{true}]

(d) $\vdash r \in (\exists v \pi(v))$

$\vdash (\pi(w) \wedge w > 0) \bigvee_{i=1}^{n} b_i$

$\vdash \pi(0) \bigvee_{i=1}^{n} ((\wedge \neg b_i) \wedge q)$

$\vdash [r] \text{fair}(\bigwedge_{i=1}^{n} [\Box c_i \rightarrow T_i])[q]$  

Here for a fair do-loop $\text{fair}(\bigwedge_{i=1}^{n} [\Box c_i \rightarrow T_i])$ only fair execution sequences are generated, i.e., finite ones or fair infinite ones, but no unfair infinite ones;

$[p]S[q]$ holds iff for all $\xi$: if input state $\xi$ satisfies $p$ then every (generated) computation sequence of $S$ in $\xi$ terminates and its output satisfies $q$;

hence $[p] \text{fair}(\bigwedge_{i=1}^{n} [\Box c_i \rightarrow T_i])[q]$ expresses that every fair computation sequence of $\bigwedge_{i=1}^{n} [\Box c_i \rightarrow T_i]$ which starts in $p$ terminates in $q$.

Note, when comparing Orna's rule with WELL, that the commands $S_j$ act as state transitions.

Since in Orna's rule the assignment $w*(D_w, S_w)$ for $w > 0$ merely generalizes WELL's notion of helpfulness function, the same kind of auxiliary quantities are required to apply both rules.
This paper proves that to express and reason about \( \mathfrak{M}, \phi \), and the assignment \( w^+(D_w, St_w) \) for \( w > 0 \) and \( w \in A \), a slight extension is required of the formalism used to prove termination of recursive procedures, Park's \( \mu \)-calculus ([6],[12]).

Finally we note that, historically, two rules have been formulated to prove termination of (nondeterministic) programs: Orna's rule ([5]) and the LPS-rule([7]). Both these rules model, each in their own way, a specific intuition related to the notion of eventuality implied by fairness assumptions. For fairly terminating loops they have been proved to be equivalent (cf. ([5])), but the LPS-rule also applies to proving fair termination of concurrent processes.

This article is organized as follows:
You are still reading chapter 1, containing the motivation for this paper; chapter 2 specifies the programming language used in this paper. Chapter 3 discusses termination under fairness assumptions. In chapter 4 the proof system and in chapter 5 the assertion-language (i.e., the monotone \( \mu \)-calculus) are dealt with. A term in the assertion-language, which expresses fair termination of a repetition is constructed in chapter 6. Completeness and soundness of Orna's rule are proven in chapters 7 and 8. Finally section 9 contains the conclusion.

Chapter 2
THE LANGUAGE OF GUARDED COMMANDS

2.1 In this chapter we describe the programming language used throughout this paper.
In section (2.2) its syntax and in section (2.3) its (relational) semantics is given.

A first-order structure \( \mathfrak{M} \) consists of (i) a non empty domain (set) \( | \mathfrak{M} | \), (ii) a set of n-ary function symbols and a set of n-ary predicate symbols \( (n \geq 2) \), such that for each n-ary function symbol (respectively predicate symbol) there corresponds a n-ary function (respectively predicate) over \( | \mathfrak{M} | \), and (iii) a set of constants, corresponding to elements of \( | \mathfrak{M} | \).
(We assume the equality symbol "=" to be present as a binary predicate symbol, corresponding to the standard equality over \( | \mathfrak{M} | \).)
2.2 SYNTAX

This paper is concerned with fair termination of repetitions. Hence for repetitions S, the notation fair(S) is introduced.

Let \( \mathcal{M} \) be some first-order structure. The language of guarded commands over \( \mathcal{M} \), \( \text{LGC}(\mathcal{M}) \), is defined by the following BNF-productions:

\[
\begin{align*}
\text{<command>} & ::= \text{<assignment>} | \text{<repetition>} | \text{<composition>} | \\
& \quad | \text{<fair loop>} \\
\text{<assignment>} & ::= \text{<variable>} \text{=}<\text{expression}> \\
\text{<composition>} & ::= \text{<guard>} \text{=}\text{<command>} \\
\text{<selection>} & ::= \{[\text{<direction>}]\} \\
\text{<direction>} & ::= \text{<guard>} \text{=}\text{<command>} \\
\text{<repetition>} & ::= \text{<selection>} \\
\text{<fair loop>} & ::= \text{fair}<\text{<repetition}> \\
\text{<expression>} & ::= "\text{term over (the signature) } \mathcal{M} " \\
\text{<guard>} & ::= "\text{quantifier-free formula over } \mathcal{M} "
\end{align*}
\]

We identify \([ \ ] \) with \( x : = x \) (skip).

In the remainder of this paper, we shall often abbreviate

\[
[b_1\circ S_1 \circ \ldots \circ b_n\circ S_n] \text{ to } \bigcirc_{i=1}^n b_i \circ S_i.
\]

2.3 SEMANTICS

A state is a function from the collection of all variables to the domain of interpretation: \( \xi, \xi', \xi_i \) etc. are used to denote states.

\( \xi(e) \) denotes the value of expression \( e \) in state \( \xi \).

If a guard \( b_i \) evaluates to true in state \( \xi \) (i.e., \( \xi(b_i) \) holds) we say that \( b_i \) is enabled in state \( \xi \). Otherwise \( b_i \) is disabled in \( \xi \).

For a variable \( x \) and an expression \( e \), \( \xi(e/x) \) is defined as usual:

\[
\begin{align*}
\xi[(e/x)](x) & = \xi(e) \\
\xi[(e/x)](y) & = \xi(y) \text{ if } x \neq y
\end{align*}
\]

Since programs depend on only finitely many variables, states can be described as functions with finite domains.

We now associate with each program \( S \) its relational semantics \( R_S: \xi \in \mathcal{M} | x | \mathcal{M} | \), where \( | \mathcal{M} | \) denotes the domain of interpretation of \( \mathcal{M} \).

Due to nondeterminism there may be more than one output-state and even infinitely many. If \( S \) nowhere terminates there will be no output-state.

\[
S = x : = e : R_S = \{ (\xi, \xi[e/x]) | \xi \text{ a state}\}
\]

\[
S = S_1 ; S_2 : R_S = R_{S_1} \circ R_{S_2}, \text{ where } \circ \text{ denotes composition of relations.}
\]

\[
S = \bigcirc_{i=1}^n [ b_i \circ S_i ] \] (n\( \geq 1 \)) : Let \( b \) denote the formula \( \gamma(b_1 \lor \ldots \lor b_n) \), and

\[
R = \bigcup_{i=1}^n R_{b_i} \circ R_{S_i}, \text{ where } R_{b_i} = \{ (\xi, \xi) | \xi \text{ a state such that } \xi(b_i) \text{ holds } \}.
\]

Then \( R_S = \bigcup_{i=1}^n R_{b_i} \circ R_{S_i} \), where \( R^1_S = \{ (\xi, \xi) | \xi \text{ a state such that } \xi(b_i) \text{ holds } \} \).

\[
S = \text{fair}(\bigcirc_{i=1}^n b_i \circ S_i) \] (n\( \geq 1 \)) : its semantics is given in chapter 3.
In the sequel we will concern ourselves exclusively with repetition statements. From now on, the term program will in general refer to a repetition.

Chapter 3
TERMINATION UNDER FAIRNESS ASSUMPTIONS

3.1 An execution sequence for a repetition \( S = \bigwedge_{i=1}^{n} b_{i} \rightarrow S_{i} \), \((n \geq 1)\) is a maximal sequence \( \xi_0, \xi_1, \xi_2, \ldots \) of states, such that \( \xi_j \vdash R_i \xi_{j+1} \), where \( j \geq 0, 1 \leq i \leq n \) and \( R_i \) is the relation associated with \( b_i \rightarrow S_i \). (The sequence is considered to be maximal if it cannot be extended, i.e., it is either infinite or the sequence is finite and ends with a state \( \xi_k \) such that \( \xi_k ( \bigwedge_{i=1}^{n} \neg b_i ) \) holds.

Termination of a (nondeterministic) program, \( S \), is straightforwardly defined as the absence of an infinite execution sequence of \( S \). This is, however, a very strong requirement.

Consider, e.g., Dijkstra's random number generator ([4]):

\[ S_0 = \# [ b+x= x+1 \land b+b= \text{false} ] \]

\( S_0 \) need not necessarily terminate if started in a state \( \xi \) such that \( \xi(b) = \text{false} \), because its execution may be governed by an extremely one-sided scheduler that consistently refuses to execute the second direction of \( S_0 \) in any iteration.

Consequently, various constraints on schedulers have been proposed, which prohibit schedulers to neglect the execution of directions under certain circumstances. Termination of a program is considered relative to a set of schedulers thus constrained. We now present two important constraints or fairness-conditions, that have been proposed ([5],[7]): fairness, and impartiality. Observe that, while the above program, \( S_0 \), admits infinite computations, none of them is fair; i.e., \( S_0 \) terminates fairly.

3.2 DEFINITION
(1) An execution sequence of a program \( S \) is fair, if it is finite or if it is infinite and every direction, which is infinitely enabled in this sequence, is chosen infinitely often.
(2) A program \( S \) terminates fairly if it admits no infinite fair execution sequences (i.e., fair(\( S \)) terminates).

In the sequel, we also need the notion of impartiality, that ignores the enabledness and disabledness of directions.

3.3 DEFINITION
(1) An execution sequence of a program is impartial, if it is finite or it is infinite and every direction occurs infinitely often in the sequence.
(2) A program terminates impartially if it admits no infinite impartial execution sequences.
The program $S_1 = \{ x = 0 \rightarrow x = x \}$ does admit infinite fair computations, but no impartial ones.

Other examples of impartially, and fairly terminating programs can be found in e.g. [5]. (Some authors use a different terminology!)

### 3.4

The relation between the two fairness assumptions is as follows:

1. for each program $S$
   1. $S$ terminates nondeterministically $\Rightarrow$
      $S$ terminates fairly
   2. $S$ terminates fairly $\Rightarrow$
      $S$ terminates impartially

The examples above show that all implications are proper.

Let $S = \{ \quad \}$ $\in [b_1 = S_1]$ (n=1). We now give the semantics $R_{\text{fair}}(S)$ associated with $\text{fair}(S)$. For each execution sequence of $\in [b_1 = S_1]$ in which $b_1$ is infinitely often enabled, $S_1$ is executed infinitely many times ($i=1,\ldots,n$).

### 3.5

Remark: definition (3.2) refers to so-called top-level fairness, according to which the following program need not terminate fairly (see e.g. [2]):

$S = \{ b_1 = \{ b_2 = \{ \}
\quad \}
\quad \}
\quad \}$

Fairness, as defined here, only constrains the choice of the directions guarded by $b_1$. It does not specify anything about choices inside the $S_1$ ($i=1,\ldots,n$). The problem of all-level fairness is not considered in this paper.
We use a Hoare-like proof system. Let $S$ be any program. By $[p]S[q]$ we mean that for all states $\xi$ satisfying $p$, the execution sequences of $S$, starting in $\xi$ are finite. Moreover every final state of such a sequence satisfies $q$.

The axioms and rules are as follows:

1. **assignment**
   
   $[p(e/x)]x := e[p]$

2. **composition**
   
   $[p]S_1[q], [q]S_2[r] \implies [p]S_1; S_2[r]$  

3. **consequence**
   
   $p \vdash p_1,$  
   \[ [p_1]S[q_1] \Rightarrow q \vdash q \]  
   \[ [p]S[q] \]

4. **Orna's rule** (see section (1.7)).

Note that only fair repetitions are considered. However, Orna's rule can also be applied to ordinary terminating do-loops (take the sets $S_t$ to be empty). We then obtain Harel's rule for terminating loops ([15]).
5.1 Our assertion-language is based on Park's monotone \( \mu \)-calculus, ([6],[12]), which is appropriate both to prove e.g. termination of recursive parameterless procedures, see e.g. [3],[6], and to express the auxiliary quantities associated with those proofs.

5.2 This calculus is based on Knaster-Tarski's theorem ([14]): let \((A,\leq)\) be a complete lattice and \(F:A\to A\) a monotonic function (in fact a cpo suffices). Then \(F\) has a least fixedpoint, denoted by \(\mu a[F(a)]\), meaning that

(i) \(F(\mu a[F(a)])=\mu a[F(a)]\);
    i.e., \(\mu a[F(a)]\) is a fixedpoint of \(F\).
(ii) if there exists some \(b\in A\) such that \(F(b)=b\),
    then \(\mu a[F(a)]\leq b\);
    i.e., \(\mu a[F(a)]\) is the least fixedpoint of \(F\).

As the partial ordering \(\leq\) is anti-symmetric, \(\mu a[F(a)]\) is unique.
(property (i) is referred to as the fixedpoint property.)

There are several ways to regard least fixedpoints. Using the notation as above, firstly \(\mu a[F(a)]=\bigwedge\{x\in A|F(x)=x\} = \bigwedge\{x\in A|F(x)\leq x\}\),
where \(\bigwedge\) denotes the infimum. A proof of this can be found in e.g. [3].
Secondly, the least fixedpoint can be obtained by iterating \(F\) into the transfinite ordinals. Define for each ordinal \(\lambda\):

\(F^0(x)=x,\)
\(F^\lambda(x)=F\left(\bigsqcup_{\beta<\lambda} F^\beta(x)\right),\) if \(\lambda \neq 0.\)

Here \(\bigsqcup\) denotes the supremum.

Let \(\downarrow A\) denotes \(A\)'s least element, which exists since \(A\) is a cpo.
Then \(\mu a[F(a)]=F^\alpha(\downarrow A)\) for some ordinal \(\alpha\). (For a proof see e.g. [9].)

Clearly, if \(\mu a[F(a)]=F^\alpha(\downarrow A)\) then for all \(\beta\geq \alpha\) \(\mu a[F(a)]=F^\beta(\downarrow A)\).

Next, we introduce some fixedpoint definitions.

Let \(R\) be a relation and \(p\) a predicate.
Define \(R\cdot p\) by \((R\cdot p)(x)\) iff \(\forall x'[(x,x')\in R\Rightarrow p(x')]\)
and its dual \(R\cdot \neg p\) by \((R\cdot \neg p)(x)\) holds iff \(\exists x'[(x,x')\in R \cdot \neg p(x')]\).
Note that \(R\cdot \text{true}\) always holds.

Since the collection of predicates ordered by \(p \leq q\) iff \(p \Rightarrow q\) forms a complete lattice with \(\text{false}\) as least element, and \(R\cdot p\) (as well as \(R\cdot \neg p\)) is monotonic in \(p\), up.\([R\cdot p]\) exists.
We claim that \( \mu^p[R^p] \) describes the domain of well-foundedness of \( R \); i.e., \( \mu^p[R^p](x) \) holds for those \( x \) such that there exists no infinite sequence \( x_0, x_1, x_2, \ldots \) with \( x = x_0 \) and \( (x_i, x_{i+1}) \in R \) (120).

\( \mu^p[R^p] = \tau^\alpha(\text{false}) \) for some ordinal \( \alpha \), where \( \tau(p) = R^p \).

Using induction on \( \beta \), we prove that for all \( \beta \in \alpha \)

\[
\tau^\beta(\text{false})(x) \Rightarrow \text{there is no infinite sequence } x_0, x_1, x_2, \ldots \\
\text{with } x = x_0 \text{ and } (x_i, x_{i+1}) \in R \text{ (120)}
\]

holds.

Induction step: \( \beta = 0 \): trivial.

Induction hypothesis: suppose that the implication holds for all \( \lambda < \beta \).

For \( \beta \neq 0 \):

\[
\tau^\beta(\text{false})(x) \iff (R \uplus \bigcup_{\lambda < \beta} \tau^\lambda(\text{false}))(x) \\
\iff \forall x' [(x, x') \in R \Rightarrow \bigcup_{\lambda < \beta} \tau^\lambda(\text{false})(x')].
\]

So \( \tau^\beta(\text{false})(x) \) implies that for all \( x' \) such that \( (x, x') \in R \) no infinite "descending" sequence starting in \( x' \) exists (induction hypothesis). Then there is no infinite "descending" sequence starting in \( x \).

To prove the other implication, assume that \( \neg \mu^p[R^p](x) \) holds (which is equivalent to \( \neg \tau^\alpha(\text{false})(x) \)).

By the fixedpoint property, \( \neg (R \uplus \mu^p[R^p]) \) (x) holds too. So, there is an \( x_1 \) such that \( (x, x_1) \in R \) and \( \neg \mu^p[R^p](x_1) \). This process can be repeated ad infinitum, and we obtain an infinite "descending" sequence \( x_0, x_1, x_2, \ldots \) such that \( x = x_0 \) and \( (x_i, x_{i+1}) \in R \) (120).

If \( F \) is a monotonic operator mapping predicates to predicates, then its greatest fixedpoint, \( \nu F(p) \), exists too. This is because the collection of predicates as defined above is a complete lattice. Moreover the greatest fixedpoint is representable in terms of the \( \nu \)-operator:

\[
\nu F(p) \iff \nu \mu^p F(p) \uplus p/p.
\]

A proof of this equivalence can be found in e.g. [3].

Using this result, we see that \( \nu R^p \) exists and that

\[
\nu R^p \iff \nu \mu^p (\neg \nu R^p)
\]

Recall that "\( \circ \)" denotes composition of relations.

We adopt the convention that "\( \circ \)" has priority over "\( \uplus \)".

I.e., \( R_1 \circ R_2 \uplus R_3 \) should be parsed as \( (R_1 \circ R_2) \uplus R_3 \).

Let \( R \) denote a relation over some set, and \( I \) the identity relation over the same set.

It is easily seen that \( F(X) = R \times X \cup I \) is monotonic in \( X \), where \( X \) denotes a relation-variable. So \( F \)'s least fixedpoint \( \mu X[R \times X \cup I] \) exists. In informal notation \( \mu X[R \times X \cup I] = I \cup R \cup R^2 \cup \ldots \cup R^n \cup \ldots \)

We abbreviate \( \mu X[R \times X \cup I] \) to \( R^* \), the relation obtained by composing \( R \), zero or more times with itself.
5.3 FACT

\[ I \subseteq R^+, R^+ \subseteq R^*, R^+ = R^* \circ R. \]

If \( T \) denotes a relation and \( T \subseteq R^* \), then \( T^* \subseteq R^* \) and \( R^* \circ T \subseteq R^* \).

5.4 Let \( \mathcal{M} \) be some first-order structure.

The first-order logic over \( \mathcal{M} \) is defined as usual. Now we extend this logic so as to be able to express fixedpoint definitions. For this an infinite set of \( n \)-ary predicate-variables, \( p, X, Y, \ldots \), is introduced for every \( n \geq 0 \). These predicate-variables may appear in formulae, but may not be bound by quantifiers. These variables form the basis of the fixedpoint definitions.

To ensure the existence of least (and greatest) fixedpoints, monotonicity has to be imposed.

In fact, we introduce the notion of syntactic monotonicity of formulae, which implies their (semantic) monotonicity. In essence, this notion requires that each occurrence of the induction-variable \( p \) is within the scope of an even number of \( \forall \)-signs.

5.5 DEFINITION

We inductively define sets \( \text{sm}(p) \), respectively, \( \text{sa}(p) \), denoting the class of formulae that are syntactically monotonic, respectively, syntactically anti-monotonic in a variable \( p \):

(i) \( \phi \in \text{sm}(p) \), if \( \phi \) does not occur free in \( \phi \).

(ii) \( \neg \phi \in \text{sm}(p) \), if \( \phi \in \text{sa}(p) \).

(iii) \( \phi_1 \lor \phi_2 \in \text{sm}(p) \), if \( \phi_1 \in \text{sa}(p) \) and \( \phi_2 \in \text{sm}(p) \).

(iv) \( \forall x \phi \in \text{sm}(p) \), if \( \phi \in \text{sm}(p) \).

(v) \( p \in \text{sm}(p) \).

(vi) \( \mu p_1.[\phi], \nu p_1.[\phi] \in \text{sm}(p) \), if \( \phi \in \text{sm}(p) \cap \text{sm}(p_1) \).

(vii) (i)-(iv) with \( \text{sm} \) and \( \text{sa} \) interchanged.

(viii) \( \mu p_1.[\phi], \nu p_1.[\phi] \in \text{sa}(p) \), if \( \phi \in \text{sa}(p) \cap \text{sm}(p_1) \).

Under the usual ordering, \( \phi_1 \leq \phi_2 \) iff \( \phi_1 \lor \phi_2 \), it can be proved by induction on the structure (complexity) of the formula that syntactic monotonicity implies semantic monotonicity.

5.6 DEFINITION

The assertion-language \( L \) over some structure \( \mathcal{M} \), is the smallest class \( B \) such that

(i) \( \phi, \mu p.[\psi(p)], \nu p.[\psi(p)] \in B \), where \( \phi \) and \( \psi \) are first-order formulae over \( \mathcal{M} \), \( \phi \) does not contain any free predicate-variables and \( \psi \in \text{sm}(p) \).

(ii) if \( \phi \in \text{B} \) then \( \phi \vee \psi, \phi \psi, \phi \psi, \neg \psi \in \text{B} \), too.

Remark: If in a formula \( \mu p.[\psi(p)] \) or \( \nu p.[\psi(p)] \) \( p \) does not occur free in \( \psi \), then we will often write \( \psi \) instead. Note that formulae of the form \( \mu p.[\psi(p)] \), where \( \psi \) contains a \( \mu \)-operator, are not allowed. However, we shall use such formulae, in which such a nesting of \( \mu \)-operators occur, since they are representable in \( L \) (see [9]).
As a well-founded set is required to apply Orna's rule, we shall need recursive ordinals. In the sequel it is assumed that there are constants for all recursive ordinals. \( \alpha, \beta, \ldots \) shall denote ordinal-constants. \( \alpha, \beta, \ldots \) are used as ordinal-variables.

The definition of validity of L-formulae is clear, except for the cases \( \text{up.}[\psi(p)] \) and \( \text{vp.}[\psi(p)] \). Recall that \( \text{up.}[\psi(p)] \) can be obtained by iteration. We now formalize this idea in the following construct by defining predicates, \( I_\psi^\beta \) for \( \beta \geq 0 \) "by iterating \( \psi \) from below".

Note that the clauses (i) and (ii) below assures us that \( I_\psi^\beta \) is monotonic in \( \beta \) and that there exists some ordinal \( k \) for which the fixedpoint is reached. In fact, \( I_\psi^k \) (as defined below) is obtained after \( k \) iterations of \( \psi \). Moreover, in this way indeed the least fixed point is obtained. This is just clause (iii) below.

To define validity of \( \text{up.}[\psi(p)] \), define predicates \( I_\psi^\beta \) for ordinals \( \beta \) by

\[
I_\psi^0, \lambda \alpha. \text{false}, I_\psi^\beta, \lambda \alpha. I_\psi^\alpha (\alpha < \beta), I_\psi^\beta = \bigcup_{\alpha < \beta} I_\psi^\alpha (\alpha \geq 0).
\]

By the monotonicity of \( \psi \) the following holds (see [9]):

(i) \( (\alpha < \beta) \Rightarrow (I_\psi^\alpha (\alpha) \Rightarrow I_\psi^\beta (\alpha)) \).

(ii) for some ordinal \( k \): \( I_\psi = I_\psi^k = \bigcup_{\alpha < k} I_\psi^\alpha \).

(iii) \( I_\psi \) is the least predicate \( C \) satisfying \( C(\alpha) \Leftrightarrow \psi(\alpha, C) \);

i.e., \( I_\psi (\alpha) \Leftrightarrow \psi(\alpha, I_\psi) \) and if \( C \) satisfies

\( C(\alpha) \Leftrightarrow \psi(\alpha, C) \) then \( I_\psi (\alpha) \Rightarrow C(\alpha) \).

We now put \( \mathcal{M} \models \text{up.}[\psi(p)] \Leftrightarrow \) for all \( \bar{x}, \mathcal{M} \models \text{up.}[\psi(p)](\bar{x}) \)

and \( \mathcal{M} \models \text{up.}[\psi(p)](\bar{x}) \Leftrightarrow I_\psi (\bar{x}) \).

Next, \( \mathcal{M} \models \text{up.}[\psi(p)] \) iff \( \mathcal{M} \models \neg \text{vp.}.\neg[\psi(p)](\neg p/p) \).

As is usual in completeness proofs, we need the ability to code finite sequences. In this case, to define the well-founded set necessary for applying Orna's-rule.

For this, we introduce the notion of acceptability of a structure ([9]).
5.8 DEFINITION

(a) A coding scheme for a set $A$ is a triple $\mathfrak{c} = <N, \leq, <>>$ such that

(i) $N^\mathfrak{c} \subseteq A$, $\leq$ is an ordering on $N$ and the structure $<N^\mathfrak{c}, \leq>$ is iso-
morphic to the integers with their usual ordering.

(ii) $<>^\mathfrak{c}$ is a one-to-one function, mapping the set $\bigcup_{n \geq 0} A^n$ of all fin-
finite sequences over $A$ to $A$.

By convention $A^0 = \emptyset$; the empty sequence $<>^\mathfrak{c}$ is the only sequence
of length 0.

(b) With each coding scheme, $\mathfrak{c}$, we associate the following
decoding relations and functions:

(i) Seq),$x^\mathfrak{c}(x)$ there exists $x_1, \ldots, x_n$ such that $x = <x_1, \ldots, x_n>^\mathfrak{c}$.

(Here, $x = <>^\mathfrak{c}$, the code of the empty sequence, is covered by the
convention that $x = <>^\mathfrak{c}$ if $n = 0$.)

(ii) The length-function, $lh^\mathfrak{c}$, for sequences maps $A$ into $N$ and hence into the integers, because of the isomorphism of $<N^\mathfrak{c}, \leq>$ with $<N, \leq>$:

$lh(x)^\mathfrak{c} =$ if $\neg \text{Seq}(x)$
$n$ if Seq$(x) \land x = <x_1, \ldots, x_n>^\mathfrak{c}$ for some $x_1, \ldots, x_n$.

(iii) The projection $(x_i)^\mathfrak{c}$ (as a function of $x$ and $i$) is defined by

$(x_i)^\mathfrak{c} = \begin{cases} x_1 \text{ if for some } x_1, \ldots, x_n, x = <x_1, \ldots, x_n>^\mathfrak{c} \text{ and } i \leq n \\ 0 \text{ otherwise} \end{cases}$

5.9 DEFINITION

A coding scheme is elementary on a structure $\mathfrak{M}$ if the relations and func-
tions $N^\mathfrak{c}, \leq, \text{Seq}, lh^\mathfrak{c}, ()$ are all elementary, i.e., first-order defin-
able on $\mathfrak{M}$.

(A function $f$ is elementary if its graph is, i.e., if $\{(x, y) | f(x) = y\}$ is
first-order definable.)

Note that the class of elementary relations on a structure is closed
under conjunction and quantification. This is an immediate consequence
of definition (5.9). It follows that the functions $p_n^\mathfrak{c}$ defined by

$p_n^\mathfrak{c}(x_1, \ldots, x_n) = <x_1, \ldots, x_n>^\mathfrak{c}$ are elementary, as

$p_n^\mathfrak{c}(x_1, \ldots, x_n) = u \iff (\text{Seq}(u) \land lh(u)^\mathfrak{c} = n \land \forall i \leq n \forall x_i (u_{\mathfrak{c} i} = x_i))$.

In the sequel we shall omit the subscripts $\mathfrak{c}$.

5.10 DEFINITION

A first-order structure $\mathfrak{M}$ is acceptable if it admits an elementary coding
scheme on $\mathfrak{M}$.

Next, we show that a number of predicates that are extensively used in
the sequel are representable in $L$. 
Let $R_1$ and $R_2$ denote relations, elementary in $\mathfrak{M}$.

The following constructs are representable in $L$:

(i) $R_1 \circ R_2$ and $R_1 \cup R_2$: this is clear.

(ii) $R_1^*$: this term is representable by $\mu X. [R_1 \circ X \cup I]$

(where $I$ denotes the identity relation).

(iii) $up. [R_1 \rightarrow p]$: define $\phi(x,p) = \forall x' [R_1(x,x') \supset p(x')]$.\(^1\)

Then $up. [\phi(x,p)]$ represents $up. [R_1 \rightarrow p] x y$

Note that this implies that $vp. [R_1 \circ p]$ is representable, too.

(iv) For predicates $r$ and relations $R$, we define a construct $r \circ R$

$r \circ R$ holds in $x$ iff there exists some $y$ satisfying $r$ and $yRx$.

("$x$ is $R$-reachable from $r$"). So $r \circ R(x) \iff \forall y[r(y) \land R(y,x)]$.

Because of (5.11) we are justified in using informal notation.

Let $\mathfrak{M}$ be a first-order acceptable structure.

For completeness, we need, amongst others, representability of the guarded commands semantics. First note that the I/O-relation of a program $S$ only constrains the valuation of its free variables (in the output-state). I.e., if $\xi S S' \xi'$ holds, then $\forall R_S \forall r \forall x \forall y$:

$\xi|X=x|X, \xi'|X'=x'|X$ and $\forall [X'=x'|X]$,

where $X$ is the set of free variables in $S$ and $|$ denotes restriction. Using this observation, the semantics is easily seen to be representable:

For example, if $S \# [b \rightarrow S']$ then $R_S(\xi, \xi') \iff \mathfrak{M} \models \mu X. [(b \circ R') \circ X \cup \neg b](x,y)$,

where $x, y$ are the codes of $\xi|B$, respectively, $\xi'|B$. (Here $R'$ denotes the relation associated with $S'$, $B$ the set of free variables occurring in $S$).

We construct an extension of $\mathfrak{M}$ by adding for every guarded command $S$ a relation-symbol $R_S$, interpreted as the semantics of $S$. Since $R_S$ is representable, we obtain a structure $\mathfrak{N}$ such that $\text{Th}(\mathfrak{M}) = \text{Th}(\mathfrak{N})$, where $\text{Th}(\mathfrak{N}) = \{ p \in \mathfrak{N} \mid \mathfrak{N} \models p \}$. I.e., $\text{Th}(\mathfrak{N})$ is conservative over $\text{Th}(\mathfrak{M})$ and we do not obtain a more expressive language in this way.

\(^1\) $(x,x') \in R$ and $R(x,x')$ are used interchangeably in this paper.
6.1 In this section we show that the property "S is fairly terminating" is representable in L.

More precisely, let \( S = \bigwedge_{i=1}^{n} b_i \rightarrow S_i \), and let \( \mathcal{M} \) be some acceptable structure. We construct a formula \( \text{FAIR}(R_1, \ldots, R_n) \) such that

\[ \mathcal{M} \models \text{FAIR}(R_1, \ldots, R_n)(\xi) \text{ iff } "S \text{ terminates fairly in } \xi" \]

Here, \( R_i \) denotes the relation associated with \( b_i; S_i \) \((i=1, \ldots, n)\).

For programs with two directions, a \( \mu \)-term expressing fair termination, has been constructed in [13].

To give the reader an idea, we construct such a term for the program \( S = \bigwedge[y > 0; x := x + 1] \bigwedge[y > 0; y := y - 1] \). This program terminates fairly. (Note that for this program fairness and impartiality coincide.)

Let \( R_1 \), respectively \( R_2 \), denote the relations associated with \( y > 0; x := x + 1 \), respectively \( y > 0; y := y - 1 \).

From section (3.2) we obtain that both \( R_1 \) and \( R_2 \), occur infinitely often in an infinite fair merge.

Now, we ask the question by what term the existence of an infinite fair sequence can be described. We consider such a sequence as consisting of an infinite number of so-called impartial parts, roughly being a finite subsequence of the infinite sequence in which every move occurs at least once.

Such an impartial part can be described as follows: \( R_1^+ \circ R_2 \cup R_2^+ \circ R_1 \).

This characterization stems from ([11]).

Remembering that truth of the predicate \( \text{vp} .[R \circ p] \) expresses the existence of an infinite sequence \( x_0, x_1, x_2, \ldots \) such that \( x_i R x_{i+1} \) for \( i \geq 0 \), the existence of an infinite fair sequence is captured by the predicate \( \text{vp} .[(R_1^+ \circ R_2 \cup R_2^+ \circ R_1) \circ p] \). Hence, program \( S \) terminates fairly in \( \xi \) iff

\[ \neg \text{vp} .[(R_1^+ \circ R_2 \cup R_2^+ \circ R_1) \circ p] \text{ (} \mu \text{p} .[(R_1^+ \circ R_2 \cup R_2^+ \circ R_1) \circ p] \text{) holds in } \xi. \]

It can be shown that this predicate holds for every \( \xi \); consequently, \( S \) terminates fairly.
6.2 IMPARTIAL TERMINATION
At first, we ignore enabledness and disabledness of directions. I.e., we

\[ \text{consider programs } \bigcap_{i=1}^{n} b \wedge S_i. \]

For such programs fairness and impartiality coincide. Assume that \( R_1, \ldots, R_n \) are the relations associated with the statements \( b; S_1, \ldots, b; S_n \) and also assume that truth of \( b \) in a state implies proper termination of \( S_i \) (\( i=1, \ldots, n \)), when started in that state.

Consequently, we first consider the problem of describing in \( L \) the existence of an infinite sequence of \( R_i \)-moves in which each of the \( R_i \) occurs infinitely often (\( i=1, \ldots, n \)).

Consider such an infinite sequence.

Since each \( R_i \) (\( i=1, \ldots, n \)) occurs an infinite number of times, this sequence may be viewed as consisting of an infinite number of finite sequences, the so-called imp(artial)parts.

Every imppart satisfies:

1. each \( R_i \) occurs in the imppart.
2. this imppart is the smallest sequence satisfying (1); i.e., any initial fragment of imppart leaves some \( R_i \) out.

To define a relation \( \text{Imppart}(R_1, \ldots, R_n) \), which expresses for every pair of states \( (\xi, \xi') \), whether \( \xi' \) can be reached from \( \xi \) by executing an imppart (w.r.t. \( R_1, \ldots, R_n \)), it suffices to consider impparts in which the first occurrences of the moves are in some prespecified order, so-called impsegments, since any imppart of \( R_1, \ldots, R_n \) is an impsegment of some permutation \( R_{i_1}, \ldots, R_{i_n} \).

More clearly, an impsegment of the ordered sequence of moves \( R_1, \ldots, R_n \) is a finite sequence in which for no \( 1 \leq i < j \leq n \) a \( R_j \)-move occurs before a \( R_i \)-move has occurred.
The relation Imppart\(R_1, \ldots, R_n\) is defined inductively (w.r.t. \(n\)) as follows:
The case \(n=1\) is simple: take Impsegment\(R_1) = R_1\).

Now, suppose that Impsegment\(R_1, \ldots, R_k\) has been defined.
Then, Impsegment\(R_1, \ldots, R_{k+1}\) looks like \(R_1, \ldots, R_i, \ldots, R_k, \ldots, R_{k+1}\), where
the first occurrences of \(R_i, R_i, R_k, R_{k+1}\) are shown (\(1 \leq i \leq k\)).

First, observe that \(R_{k+1}\) occurs only once; this is a consequence of requirement (ii) above.
Secondly, observe that the prefix \(R_1, \ldots, R_i, \ldots, R_k\) of the above sequence
is an impsegment of \(R_1, \ldots, R_k\). Hence, the sequence up to, but not including \(R_{k+1}\) is not necessarily an imppart of \(R_1, \ldots, R_k\). However, it
starts at least with an impsegment of \(R_1, \ldots, R_k\). The remaining part may
contain any (finite) number of \(R_i\)-occurrences (but no \(R_{k+1}\)).
This motivates the following definitions.

6.3 DEFINITION

Impsegment\(R_1) = R_1\)
and for \(n \geq 1:\)
Impsegment\((R_1, \ldots, R_{n+1}) = \text{Impsegment}(R_1, \ldots, R_n) \circ (R_1 \cup \ldots \cup R_n)^* \circ R_{n+1}^*\).

EXAMPLE:
Impsegment\((R_1, R_2, R_3) = R_1 \circ R_1^* \circ R_2 \circ (R_1 \cup R_2)^* \circ R_3^*\).

6.4 DEFINITION

For \(n \geq 1:\)
Imppart\((R_1, \ldots, R_n) = \bigcup_{i_1, \ldots, i_n \text{ perm of } 1, \ldots, n} \text{Impsegment}(R_{i_1}, \ldots, R_{i_n})\).

(I.e., in Imppart\((R_1, \ldots, R_n)\) the order of the \(R_i (i=1, \ldots, n)\) is immaterial.)

Remembering the example given in section (6.1), the existence of an
infinite sequence of impartial parts, starting in a state \(\xi\) is expressed
by satisfaction of a predicate Imp\((R_1, \ldots, R_n)\) in \(\xi\), defined as follows:

6.5 DEFINITION

For \(n \geq 1:\) Imp\((R_1, \ldots, R_n) = \text{vp.}[\text{Imppart}(R_1, \ldots, R_n) \circ p]\)
(Recall that \(R_i\) denote relations.)

So the program \(S = \bigcup_{i=1}^n b_i S_i\) admits an infinite fair execution se-
quence in \(\xi\) iff Imp\((R_1, \ldots, R_n)\) holds in \(\xi\).
Here \(R_i\) denotes the relation associated with \(b_i S_i (i=1, \ldots, n)\).
6.6 FAIR TERMINATION

Now, consider a program \( S = \bigcap_{i=1}^{n} \bigcap_{b_i} S_i \)

in which moves can be disabled. Assume that truth of \( b_i \) in a state implies proper termination of \( S_i \), when started in that state. Let \( R_i \) denote the relation associated with \( S_i \) (i=1,...,n).

The case of an infinite fair execution-sequence in which every move \( b_i \circ R_i \) is infinitely often enabled is easily tackled by the predicate \( \text{Imp}(b_1 \circ R_1, \ldots, b_n \circ R_n) \).

Next, suppose that move \( b_n \circ R_n \) becomes eventually never enabled anymore.

Then an infinite fair sequence of \( b_1 \circ R_1, \ldots, b_n \circ R_n \)-moves consists of some finite sequence of \( b_1 \circ R_1, \ldots, b_n \circ R_n \)-moves followed by an infinite fair sequence of \( b_1 \circ R_1, \ldots, b_n \circ R_n \)-moves in which every intermediate state satisfies \( \neg \text{b}_n \). In case no other move becomes eventually continuously disabled, this is expressed by a predicate

\[
(b_1 \circ R_1 \cup \ldots \cup b_n \circ R_n) \circ \text{Imp}(b_1 \circ \neg b_n \circ R_1, \ldots, b_{n-1} \circ \neg b_n \circ R_{n-1}).
\]

The possibility that other moves may become disabled, too, leads to the following definition:

6.7 DEFINITION

Let \( n \geq 2 \) and suppose that \( i_1, \ldots, i_n \) is some permutation of \( 1, \ldots, n \).

For \( k \), satisfying \( 1 \leq k < n \), define

\[
\text{fair}(b_{i_1} \circ R_{i_1}, \ldots, b_{i_k} \circ R_{i_k}) \text{fin}(b_{i_{k+1}} \circ R_{i_{k+1}}, \ldots, b_{i_n} \circ R_{i_n}) =
\]

\[
( \bigcup_{i=1}^{n} b_i \circ R_i) \circ \text{Imp}((b_{i_1} \circ R_{i_1}) \wedge \neg b_{i_j} \circ R_{i_j}, \ldots, (b_{i_k} \circ R_{i_k}) \wedge \neg b_{i_j} \circ R_{i_j}).
\]

REMARK:

\( \text{fair}(b_{i_1} \circ R_{i_1}, \ldots, b_{i_k} \circ R_{i_k}) \text{fin}(b_{i_{k+1}} \circ R_{i_{k+1}}, \ldots, b_{i_n} \circ R_{i_n}) \) holds in state \( \xi \) iff there exists an infinite fair sequence, starting in \( \xi \), in which the moves \( b_{i_{k+1}} \circ R_{i_{k+1}}, \ldots, b_{i_n} \circ R_{i_n} \) are eventually never enabled anymore.

Now, finally the predicate expressing the existence of infinite fair sequences can be formulated.

\[\text{1) This definition is due to P. van Emde Boas.}\]
6.8 DEFINITION

\[ \text{FAIR}(b_1 \circ R_1) = \text{Imp}(b_1 \circ R_1), \]
and for \( n \geq 2 \):

\[ \text{FAIR}(b_1 \circ R_1, \ldots, b_n \circ R_n) = \text{Imp}(b_1 \circ R_1, \ldots, b_n \circ R_n) \vee \]

\[ \bigvee_{i_1, \ldots, i_n \text{ perm of } 1, \ldots, n} \text{fair}(b_1 \circ R_1, \ldots, b_k \circ R_k) \text{fin}(b_1 \circ R_1, \ldots, b_n \circ R_n) \]

\[ 1 \leq k < n \]

In the sequel we assume that the guards \( b_i \) are incorporated in the relation \( R_i \). Also, with \( R_i \) we always associate \( b_i \) as enabling-condition.

6.9 LEMMA

\[ \mathfrak{M} \models \neg \text{FAIR}(R_1, \ldots, R_n) \iff \]

\[ [\mathfrak{M}] \models \neg \text{Imp}(R_1, \ldots, R_n) \text{ and for all } k \text{ such that } 1 \leq k < n \]

\[ \mathfrak{M} \models \bigwedge_{i_1, \ldots, i_n \text{ perm of } 1, \ldots, n} \bigwedge_{i=1}^{n} (\bigcup R_i) \neg \text{Imp}(\bigwedge_{j=k+1}^{n} \neg b_j \circ R_j, \ldots, \bigwedge_{j=k+1}^{n} \neg b_j \circ R_j). \]

PROOF

For \( n = 1 \) this follows by definition (6.8). So assume that \( n \geq 2 \). Then the lemma follows from definition (6.8), definition (6.7) and section (5.2)(the representability of the greatest fixed-point operator in terms of the least fixed-point operator).

As a last preparation for the soundness and completeness proofs, we mention the notion of the weakest precondition:

6.10 DEFINITION

An assertion \( p = \text{wlp}(S, q) \) is the weakest liberal precondition w.r.t. a command \( S \) and a condition \( q \) if

(i) \( \mathfrak{M} \models [p]S[q] \)

(ii) For each \( r \) \( \mathfrak{M} \models [r]S[q] \) implies \( \mathfrak{M} \models [p]S[q] \).

Here \( [p]S[q] \) holds iff for all \( \xi \): if input state \( \xi \) satisfies \( p \) and if \( S \) terminates, when started in \( \xi \), then each output state satisfies \( q \) (partial correctness).

Let \( S = [ \sqcup b_i \circ S_i ] \) and let \( R_i \) denotes the relation associated with \( b_i ; S_i \)

\( i = 1, \ldots, n \).

It has been shown that for each assertion \( q \) the wlp\((S, q)\) is definable in \( L \) (see [3]). It is useful to mention that in this case

\[ \text{wlp}(S, q) = (\bigcup R_i) \circ \bigwedge_{i=1}^{n} \neg b_i \circ q) \]

"if the repetition \( S \) terminates then \( q \) is satisfied in each final state, provided \( \text{wlp}(S, q) \) is satisfied at the start of the execution".
6.11 DEFINITION
An assertion \( p = \text{wp}(S,q) \) is the weakest precondition w.r.t. a command \( S \) and a condition \( q \) if
\[
\forall \mathcal{M} \models [\vdash p] \text{S}[q] \text{ and for all } r, \mathcal{M} \models [\vdash r] \text{S}[q] \text{ implies } \mathcal{M} \models [\vdash \neg p].
\]
"S always terminates in a state satisfying q, provided \( \text{wp}(S,q) \) is satisfied at the start of the execution" (total correctness).

The key theorem of this section is the following

6.12 THEOREM

For every \( \xi \): \( \text{wp}(\text{fair}(\bigwedge_{i=1}^{n} [\Box b_i \rightarrow S_i]),\xi) \iff \)
\[
\mathcal{M} \models \neg \text{FAIR}(R_1, \ldots, R_n) \land \left( \bigcup_{i=1}^{n} R_i \right)^* \land \bigwedge_{i=1}^{n} \neg b_i \rightarrow q)(\xi),
\]
where \( R_i \) are the relations associated with \( b_i \rightarrow S_i \) (\( i = 1, \ldots, n \)).

PROOF
We have to show that
\[
\mathcal{M} \models [\vdash \text{fair}(\bigwedge_{i=1}^{n} [\Box b_i \rightarrow S_i])(\xi) \iff
\]
\[
\mathcal{M} \models \neg \left( \neg \text{FAIR}(R_1, \ldots, R_n) \land \left( \bigcup_{i=1}^{n} R_i \right)^* \land \bigwedge_{i=1}^{n} \neg b_i \rightarrow q \right),
\]
" \iff Suppose that \( \mathcal{M} \models [\vdash \text{fair}(\bigwedge_{i=1}^{n} [\Box b_i \rightarrow S_i])(q) \) holds.

Choose some state \( \xi \) such that \( \mathcal{M} \models (\xi) \) holds.

Assume to obtain a contradiction that \( \mathcal{M} \models \text{FAIR}(R_1, \ldots, R_n)(\xi) \).

Then this leads immediately to a contradiction, since this implies the existence of an infinite fair execution sequence, starting in \( \xi \).

So \( \mathcal{M} \models \neg \text{FAIR}(R_1, \ldots, R_n)(\xi) \) holds.

It remains to prove that \( \mathcal{M} \models \left( \bigcup_{i=1}^{n} R_i \right)^* \land \bigwedge_{i=1}^{n} \neg b_i \rightarrow q)(\xi) \) holds, too.

To do this, choose some \( \xi' \) such that \( \mathcal{M} \models \left( \bigcup_{i=1}^{n} R_i \right)^* \land \bigwedge_{i=1}^{n} \neg b_i \rightarrow q)(\xi,\xi') \).

Clearly, then also \( \mathcal{M} \models \text{fair}(\bigwedge_{i=1}^{n} [\Box b_i \rightarrow S_i])(\xi,\xi') \), and so by the hypothesis \( \mathcal{M} \models q(\xi') \).
Suppose that $M |= \forall (\neg FAIR(R_1, \ldots, R_n) \land \bigwedge_{i=1}^{n} (\bigvee_{i=1}^{n} b_i) \land q)$. Choose state $\xi$ such that $M |= r(\xi)$.

Since, by hypothesis $M |= \neg FAIR(R_1, \ldots, R_n)(\xi)$, the repetition always terminates fairly. To prove, that in this case, each final state satisfies $q$.

Choose some $\xi'$ such that $M |= \neg FAIR(\bigwedge_{i=1}^{n} (\bigvee_{i=1}^{n} b_i))(\xi, \xi')$.

Clearly, then also $M |= \neg FAIR(R_1, \ldots, R_n)(\xi, \xi')$ and so by the hypothesis $M |= q(\xi')$ holds, which had to be shown.

6.13 COROLLARY
For every $\xi$:

$$\forall \xi \quad wp(\neg FAIR(\bigwedge_{i=1}^{n} (\bigvee_{i=1}^{n} b_i), true)) \Rightarrow
M |= \neg FAIR(R_1, \ldots, R_n)(\xi).$$

PROOF
Immediately from theorem (6.12).

This corollary states that fair termination of a repetition is indeed expressible in the $\mu$-calculus.

The remainder of this paper deals with the soundness and completeness proof of our proofsystem.

Note that the only rule for which this is non-trivial is Orna's rule.

The proof that this rule is both sound and complete is given by induction on $n$, the number of directions of the repetition. The case of only one guard is trivial, so assume that $n \geq 2$.

In the next two chapters, we prove completeness and soundness of this rule under the induction hypothesis that for all repetitions

$$S = \bigwedge_{i=1}^{k} (\bigvee_{i=1}^{n} b_i),$$

where $k < n$, and for all $r$ and $q$,

$$M |= [r]FAIR(\bigwedge_{i=1}^{k} (\bigvee_{i=1}^{n} b_i))(q) \Rightarrow Th(M) |= [r]FAIR(\bigwedge_{i=1}^{k} (\bigvee_{i=1}^{n} b_i))(q)$$

where $Th(M) = \{ p \in L \mid M \models p \}$, i.e., we assume that the theorem has been proved for syntactically simpler fair repetitions (meaning programs with less then $n$ directions).
Chapter 7

COMPLETENESS

7.1 THEOREM

Let $\mathfrak{m}$ be a first-order acceptable structure. Then our system is relative complete, meaning that for any statement $S$ and assertions $r, q \vDash L$: $\mathfrak{m} \vDash [r]S[q] \Rightarrow \text{Th}(\mathfrak{m}) \vdash [r]S[q]$, where $\text{Th}(\mathfrak{m}) = \{ p \vDash L | \mathfrak{m} \vDash p \}$. 

PROOF

The only non-trivial case is when $S \vDash \text{fair}(\forall i \leq n \bigwedge b_i \land S_i)$, and $n \geq 2$. In that case, we must show that Orna's rule can be applied. Assume that $\mathfrak{m} \vDash [r]S[q]$ holds. By theorem (6.12), we may assume, too, that $\mathfrak{m} \vDash \exists i \leq n \bigwedge (\bigvee_{i=1}^{n} \bigwedge \neg b_i) + q)$. 

At first, we must define a well-founded set $W$ and a predicate $\pi: W \times \text{States} \to \{ \text{true}, \text{false} \}$, ranking every state (reachable by $S$).

To do so, we observe that the usual approach of counting moves does not work, because not every move need to bring the program closer to termination. (E.g., in case of Dijkstra's random number generator (see section (3.1)), move $R_1$ will not help reaching termination.)

Now $S$ terminates fairly and hence also impartially (see section (3.4)). At any time, there is at least one decreasing move (otherwise there exists a state in which no move would bring the program closer to termination, resulting in the existence of an infinite fair sequence; contradiction). So, if in a successive sequence of iterations, "every enabled move has been executed at least once", then certainly the program has some other closer to termination. This shows that viewing execution-sequences as consisting of impparts, is a natural thing to do. Unfortunately, counting impparts does not work quite well, because we have to rank all states in order for Orna's rule to apply.

Consider such an imppart. It suffices that the states reached by execution this imppart, are ranked in such a way that it reflects the "progress" that is made w.r.t. executing this imppart itself.

Now a move leads to "progress" if it is a new one that has not been made in the imppart as yet. This gives the intuition behind the definitions of $W$ and $\pi$ that we now develop.

From $\mathfrak{m} \vDash \neg \text{FAIR}(R_1, \ldots, R_n) \Rightarrow \mathfrak{m} \vDash \neg \text{Imp}(R_1, \ldots, R_n)$, we obtain that $\mathfrak{m} \vDash \neg \text{FAIR}(R_1, \ldots, R_n) \Rightarrow \mathfrak{m} \vDash \text{up}.[\text{Imppart}(R_1, \ldots, R_n) \Rightarrow p]$, by applying the definitions.

As we saw in section (5.2), least fixedpoints can be obtained by iteration: Define $\tau$ by $\tau(p) = \lambda \xi . (\text{Imppart}(R_1, \ldots, R_n) \Rightarrow p)(\xi)$.

Then there exists some ordinal $\lambda$ such that $\tau^\lambda(\text{false}) = \text{up}.[\text{Imppart}(R_1, \ldots, R_n) \Rightarrow p]$. (\star)

Let $\bar{a}$ denote the least ordinal satisfying (\star).

If $\bar{a} \geq \bar{a}$ then $\tau^\bar{a}(\text{false})(\xi)$ holds for some $\xi$ iff in $\xi$ we are at most $\bar{a}$ impparts away from termination.

This gives us a way to rank the states related by impparts.
Of course, for this idea to work we need to show that \( \tau^\beta(\text{false}) \) is representable by a formula in \( L \):
(Note that \( \alpha \) is a recursive ordinal since it is less than or equal to the ordinal associated with the execution tree of \( S \), which is recursive, cf. [1]).

### 7.2 Theorem

Let \( \mathfrak{M} \) be a first-order acceptable structure.

There exists a formula \( \phi \) in \( L \) such that for all \( \xi \) and all \( \beta \leq \alpha \)

\[
\tau^\beta(\text{false})(\xi) \text{ holds iff } \mathfrak{M} \models \phi(\beta)(\xi).
\]

**Proof**

Define \( \phi(\beta) = \mu \alpha. [\exists \alpha < \beta (\text{Imppart}(R_1, \ldots, R_n) \rightarrow r(\alpha))] \). By induction on \( \beta \leq \alpha \) we prove that for all \( \beta \leq \alpha \) and all \( \xi \), \( \tau^\beta(\text{false})(\xi) \) iff \( \mathfrak{M} \models \phi(\beta)(\xi) \).

\( \beta = 0 \): trivial, since for all \( \xi \), \( \tau^{\beta}(\text{false})(\xi) \Leftrightarrow \text{false} \)

and \( \mathfrak{M} \models \phi(0)(\xi) \Leftrightarrow \mathfrak{M} \models \text{false}(\xi) \Leftrightarrow \text{false} \).

Induction hypothesis (IH): suppose that for all \( \lambda < \beta \) and all \( \xi \),

\[
\tau^\lambda(\text{false})(\xi) \text{ holds iff } \mathfrak{M} \models \phi(\lambda)(\xi).
\]

For \( \beta>0 \) we have that

\[
\mathfrak{M} \models \mu \alpha. [\exists \alpha < \beta (\text{Imppart}(R_1, \ldots, R_n) \rightarrow r(\alpha))](\xi) \text{ (definition of } \phi) \Leftrightarrow \\
\mathfrak{M} \models \exists \alpha < \beta (\text{Imppart}(R_1, \ldots, R_n) \rightarrow \phi(\alpha))(\xi) \text{ (fixedpoint property)} \Leftrightarrow \\
\text{for some } \lambda < \beta, \mathfrak{M} \models (\text{Imppart}(R_1, \ldots, R_n) \rightarrow \phi(\lambda))(\xi) \Leftrightarrow \\
\text{for some } \lambda < \beta \text{ and for all } \xi', \mathfrak{M} \models (\text{Imppart}(R_1, \ldots, R_n)(\xi, \xi') \circ \phi(\lambda)(\xi')) \Leftrightarrow \\
\text{for some } \lambda < \beta \text{ and all for } \xi',
\]

\[
\mathfrak{M} \models (\text{Imppart}(R_1, \ldots, R_n)(\xi, \xi')) \Rightarrow \tau^\lambda(\text{false})(\xi') \text{ (IH)} \Leftrightarrow \\
\text{for all } \xi', \\
\mathfrak{M} \models (\exists \lambda < \beta \tau^\lambda(\text{false})(\xi')) \Leftrightarrow \\
\text{for all } \xi', \mathfrak{M} \models (\text{Imppart}(R_1, \ldots, R_n)(\xi, \xi') \Rightarrow \bigcup_{\lambda < \beta} \tau^\lambda(\text{false})(\xi') \Leftrightarrow \\
\tau^\beta(\text{false})(\xi).
\]
Now, we define the well-founded ordered set $\mathcal{W}$ and the ranking predicate $\pi$: Each $\mathcal{W}$, not minimal, consists of two components: the first one counts imparts, the second one records "progress" within the last (incomplete) impart, and is a sequence of length at most $n$ (the number of directions of the repetition”), which records the directions within this impart, that have already been taken.

7.3 DEFINITION

$$\text{seq}_n(s) = \text{Seq}(s) \land (s_n \land \forall i[(1 \leq i \leq n) \Rightarrow (s_i, s_i)] \land \forall i,j[(1 \leq i \leq j \leq n) \land (i \neq j) \Rightarrow (s_i, s_j)].$$

(cf. definition (5.8))

Note that only directions are recorded in $\text{seq}_n$ and each direction at most once!

7.4 DEFINITION

$\mathcal{W}_a,n = \{(\bar{\lambda}, s) \mid \bar{\lambda} \subseteq \Sigma^n \land \text{seq}_n(s) \cup \{\bar{0}\}$. 

The ordering $\preceq$ defined on $\mathcal{W}_a,n$ is the following:

$\bar{0} \preceq (\bar{\lambda}, s)$ for all $(\bar{\lambda}, s) \in \mathcal{W}_a,n$ and

$$(\bar{\lambda}_1, s_1) \preceq (\bar{\lambda}_2, s_2) \iff (\bar{\lambda}_1, s_1) \preceq (\bar{\lambda}_2, s_2) \land (s_1, s_2) \land \forall i[(1 \leq i \leq n) \Rightarrow (s_1, s_2)].$$

7.5 DEFINITION

The predicate $\pi: \mathcal{W}_a,n \rightarrow (\text{States} \times \text{true, false})$ is defined by:

$$\pi(\bar{\lambda}, <) = \bar{\lambda}, \forall i (1 \leq i \leq n) \Rightarrow \pi(\bar{\lambda}, <) = \bar{\lambda}, \forall i (1 \leq i \leq n),$$

$$\pi(\bar{\lambda}, i_1, \ldots, i_k) = \bar{\lambda}, \forall i (1 \leq i \leq n),$$

$$\forall i (1 \leq i \leq n),$$

$$\forall i (1 \leq i \leq n),$$

$$\forall i (1 \leq i \leq n),$$

$$\forall i (1 \leq i \leq n),$$

$$\forall i (1 \leq i \leq n).$$

REMARK

Note that accessibility is demanded in case $\pi(\bar{\lambda}, s_1, \ldots, s_k) \land (1 \leq i \leq n)$.

If $1 \leq k < n$ and $\pi(\bar{\lambda}, s_1, \ldots, s_k) \Rightarrow (\xi)$ holds, then there exists a state $\xi'$ in which the program is at most $\bar{\lambda}$ imparts away from termination. It takes a fragment (i.e., an initial part) of an impart to reach $\xi$ from $\xi'$, namely Impart(\bar{\lambda}, \ldots, \bar{\lambda}) \Rightarrow (1 \leq j \leq n).$
Satisfaction of the clauses (a),..., (d) of Orna's rule for this choice of \( W \) and \( \pi \) follows from several lemmata and definitions, by checking that its four clauses hold indeed.

Defining \( S_0 \) and \( D_0 \) for \( w \uparrow 0 \) is simple now. If we are at the start of an impart (i.e., \( w = (\lambda, \langle \rangle \rangle) \) or \( w = (\lambda, \langle i_1, \ldots, i_n \rangle) \) for some \( \lambda \in Q \) then every move leads to eventual completion of this impart. Otherwise, \( w = (\lambda, \langle i_1, \ldots, i_k \rangle) \) for some \( \lambda, 1 \leq k < n \), and only moves different from \( R_{i_1}, \ldots, R_{i_k} \) lead to eventual completion of this impart.

7.6 DEFINITION
Let \( \omega \in \mathcal{W}_n \), \( w = (\lambda, s) \).

If \( l \uparrow h(s) = 0 \) or if \( l \uparrow h(s) = n \) then \( D_w = \{1, \ldots, n\} \) and \( S_w = \emptyset \).

If \( 0 < l \uparrow h(s) < n \) then \( D_w = \{i | (1 \leq i \leq n) \wedge \forall j \leq l \uparrow h(s)[(s) \downarrow i] \} \), \( S_w = \{1, \ldots, n\} - D_w \).

Note that for all \( \omega \in \mathcal{W}_n \), \( w \uparrow 0 : D_w \cap S_w = \emptyset \), \( D_w \neq \emptyset \) and \( D_w \cup S_w = \{1, \ldots, n\} \).

7.7 LEMMA
Let \( \omega \in \mathcal{W}_n \), \( j \in D_w \), (i.e., \( R_j \) is a decreasing move). Suppose that \( \mathfrak{M} \)

\[ \exists (\forall \text{FAIR}(R_1, \ldots, R_n) \wedge (\bigcup_{i=1}^{n} R_i)^{\pi(v)} \wedge B_{b_1})^{\pi(v)} \in \mathcal{Q} \] holds.

Then \( \text{Th}(\mathfrak{M}) \models [\pi(v) \wedge \exists j \in S_j \exists v \in \mathcal{W} \pi(v)] \) holds, too.

PROOF
We have to prove that for all \( \xi, \xi' \) states such that \( \mathfrak{M} \models R_j(\xi, \xi') \)

\[ \mathfrak{M} \models (\pi(v) \wedge \exists j \in S_j \exists v \in \mathcal{W} \pi(v)(\xi')) \] holds.

Choose states \( \xi, \xi' \) satisfying \( \mathfrak{M} \models R_j(\xi, \xi') \) and suppose that \( \mathfrak{M} \models (\pi(v) \wedge \exists j \in S_j \exists v \in \mathcal{W} \pi(v)(\xi')) \) holds.

We distinguish two cases:

(a) \( \mathfrak{M} \models \bigwedge_{i=1}^{n} \neg B_{b_i}(\xi') \).

In this case, \( \mathfrak{M} \models \pi(\langle \rangle)(\xi') \), and we are done.

---

1) Remember that \( R_j \) is the relation associated with \( b_j; S_j \).
(b) \( \mathfrak{M} \models \bigvee_{i=1}^{n} b_{i}(\xi') \).

Since \( \mathfrak{M} \models \pi(w)(\xi) \) holds, \( \mathfrak{M} \models \tau(\bigcup_{i=1}^{n} R_{i})^{\#}(\xi) \) holds, too.

I.e., \( \mathfrak{M} \models \exists \xi''[r(\xi'') \land (\bigcup_{i=1}^{n} R_{i})^{\#}(\xi'',\xi)] \).

Now \( (\bigcup_{i=1}^{n} R_{i})^{\#} \circ R_{j} \subseteq (\bigcup_{i=1}^{n} R_{i})^{\#} \) (fact (5.3)).

So it follows from \( \mathfrak{M} \models R_{j}(\xi,\xi') \) and (ii) that

\[
\mathfrak{M} \models \exists \xi''[r(\xi'') \land (\bigcup_{i=1}^{n} R_{i})^{\#}(\xi'',\xi')], \text{i.e.,}
\]
\[
\mathfrak{M} \models \tau(\bigcup_{i=1}^{n} R_{i})^{\#}(\xi').
\]

Let \( w = (\lambda, s) \). To prove \( \mathfrak{M} \models \exists v \prec w \pi(v)(\xi') \).

We distinguish three cases:

(1) \( \text{lh}(s) = 0 \), so \( s = < > \).

Since \( \mathfrak{M} \models \pi(w)(\xi) \), \( \mathfrak{M} \models \tau(\text{false})(\xi) \) holds. Consequently, it follows

that \( \mathfrak{M} \models \exists \xi''[\tau(\text{false})(\xi'') \land R_{j}(\xi'',\xi')]. \)

Hence, from \( R_{j} \subseteq R_{j}^{+} \) (fact (5.3)) we have that

\[
\mathfrak{M} \models \exists \xi''[\tau(\text{false})(\xi'') \land R_{j}^{+}(\xi'',\xi')], \text{i.e.,} \mathfrak{M} \models \tau(\text{false}) \circ R_{j}^{+}(\xi').
\]

Together with (i) and (iii), \( \mathfrak{M} \models \tau(\lambda, < >')(\xi') \) follows and hence

\( \mathfrak{M} \models \exists v \prec w \pi(v)(\xi'). \)

(2) \( 1 \leq \text{lh}(s) < n \), so \( s = < i_{1}, \ldots, i_{k} > \) for some \( i_{1}, \ldots, i_{k} \) such that \( \{i_{1}, \ldots, i_{k}\} \subseteq [1, \ldots, n] \) and \( 1 \leq k < n \). From \( \mathfrak{M} \models \pi(w)(\xi) \) we derive

\[
\mathfrak{M} \models \tau(\text{false}) \circ \text{Impsegment}(R_{i_{1}}, \ldots, R_{i_{k}}) \circ (\bigcup_{t=1}^{k} R_{i_{t}})^{\#}(\xi).
\]

Since \( \text{Impsegment}(R_{i_{1}}, \ldots, R_{i_{k}}) \circ (\bigcup_{t=1}^{k} R_{i_{t}})^{\#} \circ R_{j} \)

\[= \text{Impsegment}(R_{i_{1}}, \ldots, R_{i_{k}}, R_{j}) \]

(definition (6.3) and \( j \neq i_{1}, \ldots, i_{k} \) for \( j \in D_{w} \))

\[
\mathfrak{M} \models \tau(\text{false}) \circ \text{Impsegment}(R_{i_{1}}, \ldots, R_{i_{k}}, R_{j}) \circ (\bigcup_{t=1}^{k} R_{i_{t}} \cup R_{j})^{\#} \text{ (fact (5.3))}.
\]

This, together with the fact that \( \mathfrak{M} \models R_{j}(\xi, \xi') \) holds, implies that
\[ m \models \exists \lambda (\text{false}) \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k})_k \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 \cdot (e, \xi') \text{ holds}, \text{ too. It follows together with (i) and (iii) that } m \models \pi(\lambda, <i_1, \ldots, i_k, j>)_k(e') \]

holds.

Again, \( m \models \exists \lambda (\text{false})_k \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 \cdot (e, \xi') \) follows.

(3) \( lh(s) = n \).

From \( m \models \pi(\lambda, s)_k(e) \) and definition (7.5), the existence of a \( \exists \lambda \) such that \( m \models \pi(\lambda, s)_k(e) \) follows.

As in case (1), \( m \models \exists \lambda (\text{false})_k \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 \cdot (e, \xi') \), and so \( m \models \exists \lambda (\text{false})_k \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 \cdot (e, \xi') \).

7.8 Lemma

Let \( w \in W_\text{St}, (i.e., R_j \text{ is a steady move}). Suppose that \( m \models \lambda n \).

is a first-order acceptable structure and that

\[ m \models \exists \lambda (\text{false})_k \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 \cdot (e, \xi') \text{ holds.} \]

Then \( Th(m) \models \pi(w)_n \circ \lambda_0 \circ \lambda_1 \text{ holds, too.} \)

Proof

We have to show that for all \( \xi, \xi' \in \text{States such that } m \models R_j(e, \xi') \),

\[ m \models \pi(w)_n \circ \lambda_0 \circ \lambda_1 \text{ holds.} \]

Choose states \( \xi, \xi' \) and suppose that \( m \models \pi(w)_n \circ \lambda_0 \circ \lambda_1 \text{ holds.} \)

Let \( w = (\lambda, s) \). As in lemma (7.7) there are two cases.

\[ \begin{align*}
\text{(a)} & \quad m \models \lambda \circ \lambda_1 \circ \lambda_1 \text{,} \\
\text{Trivial.} & \\
\text{(b)} & \quad m \models \lambda \circ \lambda_1 \circ \lambda_1 \text{,} \\
\end{align*} \]

To prove \( m \models \exists \lambda (\pi(w)_n \circ \lambda_1) \text{.} \)

Note that \( lh(s) \neq 0 \) and \( lh(s) = n \), because \( \lambda n \) or \( \lambda n \) implies that \( S_{\text{St}} = s \).

So let \( w = (\lambda, <i_1, \ldots, i_k>) \), \( j, k, n \), \( (i_1, \ldots, i_k) \in [1, \ldots, n] \).

Since \( j \in \text{St}, j = i_t \) for some \( t, 1 \leq t \leq k \).

Now, \( m \models \pi(w)(e) \), so

\[ \begin{align*}
m \models (\lambda (\text{false})_k \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 '(e, \xi') \text{, i.e.,} & \\
m \models \exists \lambda '' (\lambda (\text{false})_k \circ \text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 '(e, \xi')) \text{.} & \end{align*} \]

Since \( (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 '(e, \xi') \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 '. \) (fact (5.3)), we obtain that

\[ \begin{align*}
\text{Impsegment}(R_{i_1}, \ldots, R_{i_k}) \circ (\bigcup_{t=1}^k R_{i_t})_{t=k} \circ \lambda_1 ' \text{.} & \\
\end{align*} \]
From (ii) and the fact that $\mathcal{M} = R_j(\xi, \xi')$ it then follows that

$$\mathcal{M} \models \xi'[\xi^* \text{false}] (\xi'' \wedge \text{Impsegment}(R_1, \ldots, R_k) \circ (\bigcup_{t=1}^k (\bigcup_{t=1}^k R_t)^*(\xi'', \xi'))],$$

i.e., $\mathcal{M} \models (\xi^* \text{false}) \circ \text{Impsegment}(R_1, \ldots, R_k) \circ (\bigcup_{t=1}^k R_t)^*(\xi'))$, (iii)

Moreover, as in the proof of lemma (7.7), we see that

$$\mathcal{M} \models \alpha \circ (\bigcup_{t=1}^k R_t)^*(\xi')) \quad (iv)$$

Now, (i), (iii) and (iv) imply $\mathcal{M} \models \pi \circ (\lambda, \lambda_{,1}, \ldots, \lambda_{,k})(\xi')$,

whence $\mathcal{M} \models \exists v \in w \pi(v)(\xi')$.

7.9 **LEMMA**

Suppose that $\mathcal{M} \models n \circ \text{false}(R_1, \ldots, R_n) \wedge (\bigcup_{i=1}^n R_i)^* \circ \neg v_i \circ q)$ and the induction hypothesis (i.e., for all $k$ such that $1 \leq k \leq n$

$$\mathcal{M} \models \forall_{i=1}^n (\lambda \text{fair}(\star \circ v_i \circ \epsilon_1 S_i))[q] \implies \text{Th}(\mathcal{M}) \models \forall_{i=1}^n (\lambda \text{fair}(\star \circ v_i \circ \epsilon_1 S_i))[q])$$

both hold.

Then $\text{Th}(\mathcal{M}) \models \forall_{i=1}^n (\lambda v) \wedge (\lambda v) \circ \text{false}(\star \circ v_i \circ \epsilon_1 S_i))[true]$ holds, too.

**PROOF**

Observe that for all $w \in w$ such that $w \not\models \alpha$, $D_w \not\models \psi$.

So $S_w \in [1, \ldots, n]$. It follows that the program

$S_{=\#}[\lambda v \circ \epsilon_1 S_i]$ contain less directions, than the original

$i \in S_w \quad j \in D_w$

program, so we may apply the induction hypothesis. If $S_w \not\models \psi$ then by convention $S_{=\# \text{skip}}(x := x)$, in which case the lemma is trivial. So

assume $S_w \not\models \psi$.

After a possible renumbering, we may assume, too, that $S_w \in [1, \ldots, k], \quad 1 \leq k \leq n$. So, $D_w \in [k+1, \ldots, n]$.

Let $b'$ denote $\wedge_{j \in D_w} \neg v_j = \wedge_{j \in D_w} \neg v_j$, and let $R_i = b' \circ R_i$.

By induction $\text{Th}(\mathcal{M}) \models \forall_{i=1}^n (\lambda v) \circ \text{false}(\star \circ v \circ \epsilon_1 S_i))[true]$ holds, too.

iff $\mathcal{M} \models (\lambda v) \circ \text{false}(R_1', \ldots, R_k')$ (cf. section (5.2)).

So, to prove the lemma, it suffices to show that

$\mathcal{M} \models (\lambda v) \circ \text{false}(R_1', \ldots, R_k')$.

This follows from the next two claims:
CLAIM 1
Under the aforementioned assumptions,

\[ M \models (\pi(w) \land w \land \tilde{0}) \supset \neg \text{Imp}(R_1',...,R'_k) \] holds.

**PROOF** (of claim 1)
Suppose that \[ M \models (\pi(w) \land w \land \tilde{0}) \]

Then \[ M \models \bigcup_{i=1}^{n} (\bigcup_{i=1}^{n} R_1)^*(\xi') \land \bigcup_{i=1}^{n} (\bigcup_{i=1}^{n} R_1)^*(\xi'',\xi). \]

As a consequence of our assumption, we obtain \[ M \models \neg \text{FAIR}(R_1,...,R_n) \] and so

\[ M \models \exists [\neg \text{FAIR}(R_1,...,R_n)(\xi') \land (\bigcup_{i=1}^{n} R_1)^*(\xi'',\xi)]. \] Thus,

\[ M \models \exists [\neg \text{FAIR}(R_1,...,R_n)(\xi'') \land (\bigcup_{i=1}^{n} R_1)^*(\xi'',\xi)], \]

(lemma (6.9)). Consequently,

\[ M \models \exists [\neg \text{FAIR}(R_1',...,R'_k)(\xi'') \land (\bigcup_{i=1}^{n} R_1)^*(\xi'',\xi)], \]
from which \[ M \models \neg \text{FAIR}(R_1',...,R'_k)(\xi) \] follows (definition of \( R' + p \)). \( \Box \)

Now, if \( k = 1 \), the lemma follows immediately from claim 1 and definition (6.8). So assume that \( k \geq 2 \).

CLAIM 2
Under the aforementioned assumptions,

\[ M \models (\pi(w) \land w \land \tilde{0}) \land \neg \text{fair}(R_1',...,R'_k) \land \text{fin}(R'_1,...,R'_k) \] holds.

**PROOF** (of claim 2)
Let \( 1 \leq l \leq k \). For simplicity we prove that

\[ M \models (\pi(w) \land w \land \tilde{0}) \land \neg \text{fair}(R_1',...,R'_l) \land \text{fin}(R'_l,...,R'_k) \] (any other permutation is treated in the similar way).

By definition (6.7), we must show that

\[ M \models (\pi(w) \land w \land \tilde{0}) \land (\bigcup_{i=1}^{n} R'_i)^* \land \neg \text{Imp}((\neg b' \land w' \land \neg b'_1) \land \vdots \land R'_i ... (\neg b' \land w' \land \neg b'_1) \land R'_i)) \]

holds. This is a consequence of the following chain of implications:

\[ M \models (\pi(w) \land w \land \tilde{0})(\xi) \Rightarrow \]

\[ M \models \bigcup_{i=1}^{n} (\bigcup_{i=1}^{n} R_1)^*(\xi) \] (definition (7.5)) \Rightarrow

\[ M \models \exists (\bigcup_{i=1}^{n} R_1)^*(\xi',\xi)] \] (section (5.11)) \Rightarrow
\( M \models \exists \xi' [\neg FAIR(R_1, \ldots, R_n)(\xi') \land (\bigcup_{i=1}^{n} R_1^i)^*(\xi', \xi)] \) (assumption) \( \Rightarrow \)

\( M \models \exists \xi' [(\bigcup_{i=1}^{n} R_1^i)^* \neg \text{Imp}( \land \neg b_{1,j} R_1, \ldots, \land \neg b_{j} R_1))((\xi') \land (\bigcup_{i=1}^{n} R_1^i)^*(\xi', \xi))] \).

(lemma (6.9))

Hence, for all \( t=1, \ldots, 1 \)

\[
(\neg b^i v \land \neg b_{1} R_t^i = i+1 \\
(\neg b^i v \land \neg b_{1}) b_{1} R_t^i = i+1 \\
( b^i A \land \neg b_{1} R_t^i = i+1 \\
( A R_t^i = k + 1 \\
( A A \land \neg b_{1} R_t^i = i+1 \\
( \land \neg b_{1} R_t \text{ since } 1+1 \leq k < n), i+1
\]

So (*) implies that

\[
M \models \exists \xi' [(\bigcup_{i=1}^{n} R_1^i)^* \neg \text{Imp}( (\neg b^i v \land \neg b_{1} R_t^i) \ldots (\neg b^i v \land \neg b_{1} R_t^i))((\xi') \land (\bigcup_{i=1}^{n} R_1^i)^*(\xi', \xi))] \text{ and finally} \]

\[
M \models ((\bigcup_{i=1}^{n} R_1^i)^* \neg \text{Imp}( (\neg b^i v \land \neg b_{1} R_t^i) \ldots (\neg b^i v \land \neg b_{1} R_t^i))((\xi'). \land (\bigcup_{i=1}^{n} R_1^i)^*(\xi', \xi)) \text{ (using fact (5.3)).}
\]

7.10 COROLLARY (theorem(7.1))

\[
M \models [r] \text{fair}(\{ \bigcup_{i=1}^{n} R_1^i \}^*[\bigcup_{i=1}^{n} b_{1} \cdot S_1])^[q] \Rightarrow \text{Th}(M) \models [r] \text{fair}(\{ \bigcup_{i=1}^{n} R_1^i \}^*[\bigcup_{i=1}^{n} b_{1} \cdot S_1])^[q].
\]

PROOF

From theorem(6.12), chapter 4, definition (7.4), definition (7.5), definition (7.6), lemma(7.7), lemma(7.8), lemma(7.9) and the following two observations:

(i) \( \text{Th}(M) \models [r] \exists \nu \exists (\nu) \). For let \( \xi \in \text{States} \) satisfy \( M \models [r](\xi) \).

If \( M \models \exists \nu \exists (\nu) \), then we are done, because \( M \models [r] [r] \text{fair}(\{ \bigcup_{i=1}^{n} R_1^i \}^*[\bigcup_{i=1}^{n} b_{1} \cdot S_1])^[q] \).

Hence, let \( M \models [r] \nu (\xi) \).

\[
\text{That } M \models [r] \nu (\xi) \text{ holds, follows immediately. (ii)}
\]

Since, \( M \models [r](\xi) \), also \( M \models \neg FAIR(R_1, \ldots, R_n)(\xi) \), and consequently \( M \models [r] \text{Imp}(\xi) \). (i.e., \( M \models [r] \text{false}(\xi) \))

(iii) It follows from (i), (ii), (iii) that \( M \models [r] [r] \text{false}(\xi) \).

(\xi) holds.
(ii) $\text{Th}(\mathfrak{m}) \vdash \pi(\overline{0}) \vee ((\wedge \neg b_1)^\sigma q)$.

Note that actually we showed that

$$\mathfrak{m} \models \forall (\neg \text{FAIR}(R_1, \ldots, R_n)^\sigma ((\bigcup_{i=1}^n R_i)^\sigma \wedge \neg b_1)^\sigma q) \Rightarrow$$

$$\text{Th}(\mathfrak{m}) \vdash \forall [r] \text{fair}(\pi[\circ b_1^* S_1])\pi(\overline{0}).$$

\[ (*) \]

As a consequence of the hypothesis,

$$\mathfrak{m} \models \forall ((\bigcup_{i=1}^n R_i)^\sigma \wedge \neg b_1)^\sigma q).$$

So, by (*) $\mathfrak{m} \models \pi(\overline{0}) \vee q$. Now, $\text{Th}(\mathfrak{m}) \vdash \pi(\overline{0}) \vee q$ follows.

Chapter 8
SOUNDNESS

Soundness of Orna's rule amounts to the following

8.1 THEOREM

Let $\mathfrak{m}$ be a first-order acceptable structure. Then

$$\text{Th}(\mathfrak{m}) \vdash [r] \text{fair}(S)[q] \Rightarrow \mathfrak{m} \models [r] \text{fair}(S)[q], \text{ where } S^{\pi}[\circ b_1^* S_1] (n \geq 1),$$

PROOF

Again, the non-trivial case is when $n \geq 2$.

Assume that $\text{Th}(\mathfrak{m}) \vdash [r]S[q]$.

By theorem (6.12) it suffices to show that

$$\mathfrak{m} \models \forall (\neg \text{FAIR}(R_1, \ldots, R_n)^\sigma ((\bigcup_{i=1}^n R_i)^\sigma \wedge \neg b_1)^\sigma q) \text{ holds, where } R_i \text{ is the relation associated with } b_1^* S_1 (i=1, \ldots, n).$$

Let $W$ and $\pi$ be the well-founded set, respectively, the ranking function, that where used when applying Orna's rule.

8.2 LEMMA

Assume that $\text{Th}(\mathfrak{m}) \vdash [r] \text{fair}(*[\circ b_1^* S_1])[q]$ holds.

Then $\mathfrak{m} \models \forall \pi \text{Imp}(R_1, \ldots, R_n) \text{ holds, too}.$

PROOF

Let $\mathfrak{m} \models (\xi)$ and suppose, to obtain a contradiction, that

$$\mathfrak{m} \models \text{Imp}(R_1, \ldots, R_n)(\xi) \text{ holds. Since } D_{w^*} \forall w^0 \text{ for } w \geq 0, \text{ there exists an infinite decreasing sequence in } W, \text{ starting in some } w \in W \text{ such that } \mathfrak{m} \models \pi(w)(\xi) \text{ holds. This contradicts the well-foundedness of } W.$$
8.3 **Lemma**

Assume that \( \text{Th}(\mathcal{M}) \models [r] \text{fair}(*[\Box b_i \rightarrow S_i]) [q] \) holds.

Let \( k \) be given, \( 1 \leq k \leq n \), and assume furthermore that \( i_1, \ldots, i_n \) is some permutation of \( 1, \ldots, n \) (\( n \geq 2 \)).

Then \( \mathcal{M} \models [\forall R_i, \ldots, R_k \rightarrow \text{fin}](R_{i_1}, \ldots, R_{i_{k+1}}, \ldots, R_n) \) holds, too.

**Proof**

Assume that \( \mathcal{M} \models r(\xi) \) holds for some \( \xi \).

Possibly, after a renumbering, let \( i_1, \ldots, i_n \) be the identity permutation of \( 1, \ldots, n \). Hence, we show that

\[ \mathcal{M} \models [\forall R_i, \ldots, R_k \rightarrow \text{fin}](R_{i_{k+1}}, \ldots, R_n)(\xi) \] holds. According to definition (6.7), it suffices to prove the following

**Claim**

For all \( \xi' \) satisfying \( \mathcal{M} \models (\bigcup_{i=1}^{n} R_i)^{*}(\xi, \xi') \),

\[ \mathcal{M} \models [\forall R_i \rightarrow \text{Imp}(b_{i} \circ R_1, \ldots, b_{i} \circ R_k)(\xi') \] holds, where \( b_{i} = \bigwedge_{i=k+1}^{n} b_{i} \).

**Proof (of the claim)**

Assume this is false. Both \( \xi \) and \( \xi' \) are accessible states; i.e., both

\[ \mathcal{M} \models [\forall R_i \rightarrow \text{Imp}(\bigcup_{i=1}^{n} R_i)^{*}(\xi)] \text{ and } \mathcal{M} \models [\forall R_i \rightarrow \text{Imp}(\bigcup_{i=1}^{n} R_i)^{*}(\xi')] \] hold.

From our assumption that \( \mathcal{M} \models [\forall R_i \rightarrow \text{Imp}(b_{i} \circ R_1, \ldots, b_{i} \circ R_k)(\xi') \) holds, we infer the existence of an infinite fair sequence of moves \( b_{i} \circ R_1, \ldots, b_{i} \circ R_k \). As a consequence of our assumption that \( \text{Th}(\mathcal{M}) \models [\forall R_i \rightarrow \text{fair}(*[\Box b_i \rightarrow S_i]) [q] \) holds,
we conclude that Orna's rule has been applied. Consequently, related to this sequence of moves, is an infinite sequence \( w_1, w_2, w_3, \ldots \) in \( W \) such that \( w_i \not= w_{i+1} \) and \( \mathcal{M} \models [\forall w_i (\xi')] \). Since \( W \) is well-founded, eventually \( w_i = w_{i+1} \).

This implies that none of the moves eventually taken, are decreasing moves.

Furthermore there is a state \( \xi'' \) such that

(a) \( \mathcal{M} \models [\text{Imp}(\forall R_1, \ldots, b_{i} \circ R_k)]^{*}(\xi', \xi'') \),

(b) \( \mathcal{M} \models [\forall R_i \rightarrow \text{Imp}(b_{i} \circ R_1, \ldots, b_{i} \circ R_k)(\xi'')] \), and

(c) there is a \( w'' \) (not minimal) satisfying \( w'' \not= w' \), \( \mathcal{M} \models [\forall w''(\xi'')] \) and \( \{1, \ldots, k\} \subseteq \text{St}_{w''} \).

Let \( \text{St}_{w''} = \{j_1, \ldots, j_{k+m}\} \) for some \( m \geq 0 \), where \( j_{t} = t \) for \( t = 1, \ldots, k \).

(\( \text{so } D_{w''} = \{j_{k+m+1}, \ldots, j_n\} \)). Now, \( w'' \not= 0 \) and
Thus (\(\mathfrak{M}\) : [\(\pi(w')\) \(\land w' > 0\)] \(\downarrow\) \(\mathcal{H}\) \((\mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downarrow \mathcal{J} \downaw}