The calculus of SPECTRUM*

Franz Regensburger

July 29, 1994

Abstract

In this paper I present the logical calculus for the specification language SPECTRUM [BFG+93a, BFG+93b]. It is a three-valued variant of the calculus for the logic LCF [Pau87]. Familiarity with the technical report [GR94] in which the semantics of SPECTRUM is defined is assumed.

---

*This work is sponsored by the German Bundesministerium für Forschung und Technik (BMFT) as part of the compound project "KORSO -Korrekte Software".

Contents

1 Introduction

In this paper I present the logical calculus for the specification language SPECTRUM [BFG+93a, BFG+93b]. The definition of the syntax and semantics of the core language of SPECTRUM can be found in [GR94]. Familiarity with this technical report is assumed since I will use the notation introduced in this paper.

The paper is organized as follows. In section 2 sequents and their semantics are introduced. In Section 3 the inference rules of the calculus are presented and in section 4 I will show some selected derived rules. The paper is closed by some concluding remarks about the SPECTRUM logic in section 5. Although the calculus was formalized in the logical framework Isabelle [Pau93a] and all derived rules have been derived in this framework I will not present the Isabelle formalization in this paper.

2 Sequents

In this section I introduce the notion of a sequent. Sequents are used to formalize the logical calculus of SPECTRUM as a natural deduction system with explicit treatment of living hypotheses. This makes the formalization of side conditions for rules much easier and allows to reason directly about the correctness of rules and of the overall calculus. This kind of formalization is well known from other logical systems, e.g. the HOL logic [GM93]. The definition of sequent is due to [GM93] but is adjusted to three-valued logic.

In order to simplify the following definitions I will make a slight change in terminology with respect to the one introduced in [GR94].

Definition 1 *well-formed terms, formulae*

Fixing a signature Σ and a family of type variables $\mathcal{X}$, a well-formed term of sort τ is a pre-term e, such that there exists a sort derivation that ends with $\Gamma \rhd_{\mathcal{X}} \ e :: \tau$. The set T_τ of well-formed terms of sort τ is thus defined as:

$$T_\tau = \{e \mid \text{ there exists } \Gamma \text{ with } \Gamma \rhd_{\mathcal{X}} \ e :: \tau\}$$

Formulae are terms of sort Bool:

$$\mathsf{Form} = T_{\mathsf{Bool}}$$

The tuples $(\mathcal{X}, \Gamma, e, \tau)$ that are called well-formed terms in [GR94] are called *terms in context* in this paper.

If the type of the term e is known, there exists a unique minimal variable context Γ consisting just of type assumptions for all the variables occuring free in e. This is due to the fact that terms are almost fully typed. If the term e is not a variable then one even doesn't need to know the outermost type of e to construct the unique and minimal variable context Γ. This means that for every formula $p \in \mathsf{Form}$, where the type is known to be Bool, there exists a unique minimal context Γ such that $(\mathcal{X}, \Gamma, e, \mathsf{Bool})$ is a term in context and there is also a unique normal sort derivation that ends with $\Gamma \rhd_{\mathcal{X}} \ e :: \mathsf{Bool}$.

Definition 2 *Sequents*

Fixing a signature Σ and a family of type variables $\mathcal{X}$, a sequent is a pair (H, p) where $H \subseteq \mathsf{Form}$ is a finite set of formulae and $p \in \mathsf{Form}$ is a formula. The set H is called the set of assumptions and p is called the conclusion of the sequent[1]. Instead of (H, p) I will use the notation $H \blacktriangleright p$ for sequents and list notation $h_1, \ldots, h_n$ or H, h_1 instead of $\{h_1, \ldots, h_n\}$ or $H \cup \{h_1\}$.

There is one additional property that must hold for a sequent $h_1, \ldots, h_n \blacktriangleright p$ in order to give it a semantics later on. We require:

$$h_1 \wedge \ \ldots \wedge h_n \wedge p \in \mathsf{Form}$$

If the condition is fulfilled there is a unique and common type assumption for every variable occuring free in the terms $h_1, \ldots, h_n$ and p. The use of the $\wedge$ connective here is arbitrary. Its only purpose is to connect the h_i and p to form a single term.

Usually the semantics of a sequent

$$h_1, \ldots, h_n \blacktriangleright p$$

is defined to be that of the formula

$$h_1 \wedge \ \ldots \wedge h_n \Rightarrow p$$

This means that a sequent is to be red as implication. Since SPECTRUM uses a three-valued logic, this definition needs to be changed. There are several opportunities to extend the semantics of a sequent from the two-valued to the three-valued case. Olaf Owe gives a detailed survey in [Owe91] about this topic. In his terminologies the set of assumptions H and the conclusion p both have a strong interpretation in SPECTRUM. Therefore the semantics of a sequent

$$h_1, \ldots, h_n \blacktriangleright p$$

is defined to be that of the formula

$$h_1 \wedge \delta(h_1) \wedge \ \ldots \wedge h_n \wedge \delta(h_n) \Rightarrow p$$

Owe would characterize the partial logic of SPECTRUM with SS in his tables. Using the notation of [GR94] the semantics of a sequent is defined as follows:

[1] historically, a sequent (in German 'Sequenz' due to [Gen35]) is a list and not a set. The use of lists is vital for real sequent calculi like Gentzen's LK. In this paper the assumptions are the set of living hypotheses in a natural deduction calculus and therefore the order and multiplicity of formulae in the sequent is not interesting. It is a terminological accident that the pair (H, p) is called a sequent but this terminology is used by several authors [Pau87, GM93].

Definition 3 *Satisfaction for sequents*

Given a signature Σ, a family of type variables $\mathcal{X}$ and a Σ-Algebra A, the sequent $h_1, \ldots, h_n \blacktriangleright p$ is called valid in A rsp. A satisfies $h_1, \ldots, h_n \blacktriangleright p$
(in symbols: $A \models h_1, \ldots, h_n \blacktriangleright p$) iff:

$$(A \models (\mathcal{X}, \Gamma, h_1 \wedge \delta(h_1) \wedge \ \ldots \ \wedge h_n \wedge \delta(h_n) \Rightarrow p, \mathsf{Bool})$$

where Γ is the unique variable context.

The definitions for a deductive system and for a formal proof with respect to a deductive system are that of [GM93] and will not be repeated here.

3 Inference Rules

The inference rules of the logical calculus of SPECTRUM are given in the style of [GM93]. Using this style the translation into the Isabelle system is straight forward. Any side conditions restricting the applicability of a rule are given to the right of it. The names in square brackets are the names of the rules in the Isabelle formalization.

For the notation of terms I do not use the rigid syntax of [GR94] but use a more sloppy one with infix notation. Also type information is suppressed whenever an order-sorted type inference mechanism in the sense of [Nip91] could infer the missing types. Actually the untyped terms presented below were also used in the Isabelle formalization where Isabelle's type inference machine computed the most general type information for the missing types.

3.1 Propositional logic

[hyp] $\dfrac{}{H \blacktriangleright p}\Big\{ \ p \in H$

[weak] $\dfrac{H_1 \blacktriangleright p}{H_2 \blacktriangleright p}\Big\{ \ H_1 \subseteq H_2$

The rules [hyp] and [weak] are covered by Isabelle's meta-logic.

[exmid] $\dfrac{H \blacktriangleright \delta(p)}{H \blacktriangleright p \vee \neg p}$

[conjI] $\dfrac{H_1 \blacktriangleright p \quad H_2 \blacktriangleright q}{H_1, H_2 \blacktriangleright p \wedge q}$

[conjunct1] $\dfrac{H \blacktriangleright p \wedge q}{H \blacktriangleright p}$

[conjunct2] $\dfrac{H \blacktriangleright p \wedge q}{H \blacktriangleright q}$

[disjI1] $\dfrac{H \blacktriangleright p}{H \blacktriangleright p \vee q}$

[disjI2] $\dfrac{H \blacktriangleright q}{H \blacktriangleright p \vee q}$

[disjE] $\dfrac{H_1 \blacktriangleright p \vee q \quad H_2, p \blacktriangleright r \quad H_3, q \blacktriangleright r}{H_1, H_2, H_3 \blacktriangleright r}$

[impI] $\dfrac{H_1 \blacktriangleright \delta(p) \quad H_2 \blacktriangleright q}{(H_1 \cup H_2) \setminus \{p\} \blacktriangleright p \Rightarrow q}$

[mp] $\dfrac{H_1 \blacktriangleright p \Rightarrow q \quad H_2 \blacktriangleright p}{H_1, H_2 \blacktriangleright q}$

[not_def] $\dfrac{}{\emptyset \blacktriangleright \neg p = (p \Rightarrow \mathsf{false})}$

[FF_E] $\dfrac{H \blacktriangleright \mathsf{false}}{H \blacktriangleright p}$

3.2 Embedding into Bool

[TT_I] $\dfrac{H \blacktriangleright p}{H \blacktriangleright p = \mathsf{true}}$

[TT_E] $\dfrac{H \blacktriangleright p = \mathsf{true}}{H \blacktriangleright p}$

[eq_to_imp] $\dfrac{H_1 \blacktriangleright p = q \quad H_2 \blacktriangleright \delta(p)}{H_1, H_2 \blacktriangleright (p \Rightarrow q) \wedge (q \Rightarrow p)}$

[impl_to_eq] $\dfrac{H \blacktriangleright (p \Rightarrow q) \wedge (q \Rightarrow p)}{H \blacktriangleright p = q}$

3.3 Quantifiers

[ALLbI] $\dfrac{H \blacktriangleright p}{H \blacktriangleright \forall^{\perp} x.p}$ provided $x \notin FV(H)$

[ALLbE] $\dfrac{H \blacktriangleright \forall^{\perp} x.p}{H \blacktriangleright p[t/x]}$

In the rule above $p[t/x]$ denotes the substitution of the term t of appropriate type for the variable x in formula p, with suitable renaming of bound variables to prevent free variables in t becoming bound after substitution.

[EXbI] $\dfrac{H \blacktriangleright p[t/x]}{H \blacktriangleright \exists^{\perp} x.p}$

[EXbE] $\dfrac{H_1 \blacktriangleright \exists^{\perp} x.p \quad H_2 \blacktriangleright p \Rightarrow q}{H_1, H_2 \blacktriangleright q}$ { provided $x \notin (FV(H_1) \cup FV(H_2) \cup FV(q))$

[DEF_ALLb] $\dfrac{}{\emptyset \blacktriangleright \delta(\forall^{\perp} x.p) = ((\forall^{\perp} x.p = \mathsf{true}) \vee (\exists^{\perp} x.p = \mathsf{false}))}$

[DEF_EXb] $\dfrac{}{\emptyset \blacktriangleright \delta(\exists^{\perp} x.p) = ((\exists^{\perp} x.p = \mathsf{true}) \vee (\forall^{\perp} x.p = \mathsf{false}))}$

[ALL_def] $\dfrac{}{\emptyset \blacktriangleright (\forall x.p) = (\forall^{\perp} x.\delta(x) \Rightarrow p)}$

[EX_def] $\dfrac{}{\emptyset \blacktriangleright (\exists x.p) = (\exists^{\perp} x.\delta(x) \wedge p)}$

3.4 Equality

[beta_red] $\dfrac{}{\emptyset \blacktriangleright (\lambda x.e)t = e[t/x]}$

[subst] $\dfrac{H_1 \blacktriangleright t_1 = t_2 \quad H_2 \blacktriangleright p[t_1/x]}{H_1, H_2 \blacktriangleright p[t_2/x]}$

In the rule above $p[t_1/x]$ means that p is a formula in which the variable x eventually occurs and that the term t_1 is substituted for every free occurrence of x with usual prevention of variable capture.

3.5 Domain Theory

[refl_less] $\dfrac{}{\emptyset \blacktriangleright x \sqsubseteq x}$

[eq_def] $\dfrac{}{\emptyset \blacktriangleright (x = y) = (x \sqsubseteq y \wedge y \sqsubseteq x)}$

[trans_less] $\dfrac{H_1 \blacktriangleright x \sqsubseteq y \quad H_2 \blacktriangleright y \sqsubseteq z}{H_1, H_2 \blacktriangleright x \sqsubseteq z}$

[ext_less] $\dfrac{H_1 \blacktriangleright \delta(g) \quad H_2 \blacktriangleright \forall^{\perp} x.f(x) \sqsubseteq g(x)}{H_1, H_2 \blacktriangleright f \sqsubseteq g}\left\{ x \notin FV(f) \cup FV(g) \right.$

[mono_less] $\dfrac{H_1 \blacktriangleright f \sqsubseteq g \quad H_2 \blacktriangleright x \sqsubseteq y}{H_1, H_2 \blacktriangleright f(x) \sqsubseteq g(y)}$

[minimal] $\dfrac{}{\emptyset \blacktriangleright \perp \sqsubseteq x}$

[DEF_app] $\dfrac{}{\emptyset \blacktriangleright \perp(x) = \perp}$

[DEF_def] $\dfrac{}{\emptyset \blacktriangleright \delta(x) = \neg(x = \perp)}$

[fix_eq] $\dfrac{}{\emptyset \blacktriangleright f(\mathsf{fix}(f)) = \mathsf{fix}(f)}$

[strong_less] $\dfrac{}{\emptyset \blacktriangleright \delta(x \sqsubseteq y)}$

[strong_eq] $\dfrac{}{\emptyset \blacktriangleright \delta(x = y)}$

[strong_DEF] $\dfrac{}{\emptyset \blacktriangleright \delta(\delta(x))}$

[DEF_LAM] $\dfrac{}{\emptyset \blacktriangleright \delta(\lambda x.t)}$

3.6 Polymorphism

The following rule is already part of Isabelle's meta-logic

[inst] $\dfrac{H \blacktriangleright p}{\sigma(H) \blacktriangleright \sigma(p)}\left\{ \text{where } \sigma \text{ is an order-sorted type substitution} \right.$

3.7 Axioms for the datatype Bool

[DEF_TT] $\dfrac{}{\emptyset \blacktriangleright \delta(\mathsf{true})}$

[DEF_FF] $\dfrac{}{\emptyset \blacktriangleright \delta(\mathsf{false})}$

[DEF_conj] $\dfrac{}{\emptyset \blacktriangleright \delta(\ \wedge\ )}$

[DEF_disj] $\dfrac{}{\emptyset \blacktriangleright \delta(\ \vee\ )}$

[DEF_impl] $\dfrac{}{\emptyset \blacktriangleright \delta(\ \Rightarrow\ )}$

[DEF_not] $\dfrac{}{\emptyset \blacktriangleright \delta(\neg)}$

[conj_ax1] $\dfrac{}{\emptyset \blacktriangleright (x \wedge y) = (y \wedge x)}$

[conj_ax2] $\dfrac{}{\emptyset \blacktriangleright (\mathsf{true} \wedge y) = y}$

[conj_ax3] $\dfrac{}{\emptyset \blacktriangleright (\mathsf{false} \wedge y) = \mathsf{false}}$

[conj_ax4] $\dfrac{}{\emptyset \blacktriangleright (\bot \wedge \bot) = \bot}$

[disj_ax1] $\dfrac{}{\emptyset \blacktriangleright (x \vee y) = (y \vee x)}$

[disj_ax2] $\dfrac{}{\emptyset \blacktriangleright (\mathsf{true} \vee y) = \mathsf{true}}$

[disj_ax3] $\dfrac{}{\emptyset \blacktriangleright (\mathsf{false} \vee y) = y}$

[disj_ax4] $\dfrac{}{\emptyset \blacktriangleright (\bot \vee \bot) = \bot}$

[impl_ax1] $\dfrac{}{\emptyset \blacktriangleright (\mathsf{true} \Rightarrow y) = y}$

[impl_ax2] $\dfrac{}{\emptyset \blacktriangleright (\mathsf{false} \Rightarrow y) = \mathsf{true}}$

[impl_ax3] $\dfrac{}{\emptyset \blacktriangleright (\bot \Rightarrow \mathsf{true}) = \mathsf{true}}$

[impl_ax4] $\dfrac{}{\emptyset \blacktriangleright (\bot \Rightarrow \mathsf{false}) = \bot}$

[impl_ax5] $\dfrac{}{\emptyset \blacktriangleright (\bot \Rightarrow \bot) = \bot}$

[Exh_BOOL] $\dfrac{}{\emptyset \blacktriangleright x = \mathsf{true} \vee x = \mathsf{false} \vee x = \bot}$

[not_FF_less_UU] $\dfrac{}{\emptyset \blacktriangleright \neg(\mathsf{false} \sqsubseteq \bot)}$

[not_FF_less_TT] $\dfrac{}{\emptyset \blacktriangleright \neg(\mathsf{false} \sqsubseteq \mathsf{true})}$

[not_TT_less_UU] $\dfrac{}{\emptyset \blacktriangleright \neg(\mathsf{true} \sqsubseteq \bot)}$

[not_TT_less_FF] $\dfrac{}{\emptyset \blacktriangleright \neg(\mathsf{true} \sqsubseteq \mathsf{false})}$

3.8 Tuples

The following rules are schemas that can be instantiated for every $n \geq 2$. However, in the Isabelle formalization these rules are only partially implemented for $2 \leq n \leq 5$.

[beta_red$_n$] $\dfrac{}{\emptyset \blacktriangleright (\lambda\langle x_1, \ldots, x_n\rangle.e)\langle t_1, \ldots, t_n\rangle = e[t_1/x_1 \ldots t_n/x_n]}$

The term $e[t_1/x_1 \ldots t_n/x_n]$ denotes simultaneous substitution of the terms t_i for x_i.

[DEF_LAM$_n$] $\dfrac{}{\emptyset \blacktriangleright \delta(\lambda\langle x_1, \ldots, x_n\rangle.t)}$

[Exh_prod$_n$] $\dfrac{}{\emptyset \blacktriangleright \forall^{\perp} z.\exists^{\perp} x_1 \ldots x_n.\langle x_1, \ldots, x_n\rangle = z}$

3.9 Fixed point induction

[fix_ind] $\dfrac{H_1 \blacktriangleright \forall^{\perp} x.\, \delta(p) \quad H_2 \blacktriangleright p[\perp/x] \quad H_3 \blacktriangleright \forall^{\perp} x.\, p \Rightarrow p[f(x)/x]}{H_1, H_2, H_3 \blacktriangleright p[\mathsf{fix}(f)/x]} \left\{ p \ddagger x \right.$

Note that due to the first premise induction is only allowed for formulae that are defined everywhere. In the above rule $p \ddagger x$ (pronounce 'p double-dagger x') means that the formula p must be admissible in x. There are well known (sufficient) properties [Pau87] to check the admissibility of a formula p in a free variable x. However, since in SPECTRUM the terms are not necessarily continuous in all their free variables, the test is a bit more complicated and interconnected with a test for continuity.

3.9.1 Test for continuity

In [GR94] we presented a very tricky syntactical test, the †-test (pronounce 'dagger test') to test the continuity of a term in a certain variable allowing this term to contain mapping symbols and also quantifiers in a restricted form. This test was needed to formulate a context condition for the formation rules (abstr) and (patt-abstr) concerning λ-abstraction in order to guarantee the type-correctness of our formation calculus. The two rules of [GR94] are given below:

$$(\text{abstr})\frac{\Gamma, x:\tau_1 \rhd_\chi e :: \tau_2}{\Gamma \rhd_\chi \lambda x:\tau_1.e :: \tau_1 \to \tau_2}\,\{e \dagger x$$

$$(\text{patt-abstr})\frac{\Gamma, x_1:\tau_1, \ldots, x_n:\tau_n \rhd_\chi e :: \tau}{\Gamma \rhd_\chi \lambda\langle x_1:\tau_1, \ldots, x_n:\tau_n\rangle.e :: \tau_1 \times \ldots \times \tau_n \to \tau}\left\{\begin{array}{l} e \dagger x_i \\ 1 \le i \le n \end{array}\right.$$

However, in practice it turned out that in order to preserve continuity, the †-test had to restrict the use of mapping symbols and quantifiers in a way that they could only occure in positions where they are of little use. Therefore we decided to be more restrictive again and formulated a new context condition which is also implemented in the current analyser for the SPECTRUM language. The new context condition $\mathsf{cont}(e)$ qualifies a term e to be continuous *in all* its free variables if e does not contain any mapping symbols or quantifiers.

$$\frac{e \text{ contains no mappings or quantifiers}}{\mathsf{cont}(e)}$$

If e does not contain any mapping symbols or quantifiers there must be a sort derivation for e that uses only the formation rules (var), (const), (Π-inst), (weak), (tuple), (abstr), (patt-abstr) and (appl)[2]. It is well known, e.g. [Sch86], that such a term e is continuous in all its free variables. The new formation rules for abstraction are now:

$$(\text{abstr})\frac{\Gamma, x:\tau_1 \rhd_\chi e :: \tau_2}{\Gamma \rhd_\chi \lambda x:\tau_1.e :: \tau_1 \to \tau_2}\,\{\mathsf{cont}(e)$$

$$(\text{patt-abstr})\frac{\Gamma, x_1:\tau_1, \ldots, x_n:\tau_n \rhd_\chi e :: \tau}{\Gamma \rhd_\chi \lambda\langle x_1:\tau_1, \ldots, x_n:\tau_n\rangle.e :: \tau_1 \times \ldots \times \tau_n \to \tau}\,\{\mathsf{cont}(e)$$

3.9.2 Test for admissibility ‡

The property $e \ddagger x$ is recursively defined on the structure of the well-formed term e. It's reading is 'e double-dagger x' and means 'e is admissible in x'.

Axioms:

$$(\ddagger\text{not\_free})\frac{}{t \ddagger x}\left\{\text{ provided } x \notin FV(t)\right.$$

[2] see [GR94] for details.

$$(\ddagger\neg\bot\sqsubseteq)\frac{}{\neg(\bot\sqsubseteq u)\ddagger x}$$

Rules that need continuity:

$$(\ddagger\sqsubseteq)\frac{\mathsf{cont}(t)\quad\mathsf{cont}(u)}{t\sqsubseteq u\ddagger x}$$

$$(\ddagger\neg\sqsubseteq)\frac{\mathsf{cont}(t)}{\neg(t\sqsubseteq u)\ddagger x}\Big\{\text{provided } x\notin FV(u)$$

$$(\ddagger\mathrm{subst})\frac{\mathsf{cont}(t)\quad p\ddagger x}{p[t/x]\ddagger x}$$

$$(\ddagger\neg=\bot)\frac{\mathsf{cont}(t)}{\neg(t=\bot)\ddagger x}$$

$$(\ddagger=)\frac{\mathsf{cont}(t)\quad\mathsf{cont}(u)}{t=u\ddagger x}$$

Propagation of admissibility:

$$(\ddagger\wedge)\frac{p\ddagger x\quad q\ddagger x}{p\wedge q\ddagger x}$$

$$(\ddagger\vee)\frac{p\ddagger x\quad q\ddagger x}{p\vee q\ddagger x}$$

$$(\ddagger\Rightarrow)\frac{\neg p\ddagger x\quad q\ddagger x}{p\Rightarrow q\ddagger x}$$

$$(\ddagger\forall^{\bot})\frac{p\ddagger x}{\forall^{\bot}y.\,p\ddagger x}$$

In addition to these syntactical rules there are two more criteria for admissibility. They can be expressed in our formalism in a somehow crude way by mixing the $\ddagger$-test and the logical calculus.

The first one is about equivalent formulae and means that every formula that is equivalent to an admissible formula is also admissible:

$$(\ddagger\mathrm{cong})\frac{\emptyset\blacktriangleright\forall^{\bot}x.\,p=q\quad p\ddagger x}{q\ddagger x}$$

The second is about flat types and means that every predicate over a flat type is admissible:

$$(\ddagger\text{flat})\ \frac{\emptyset \blacktriangleright \forall^{\perp} y{:}\tau.\forall^{\perp} z{:}\tau.\ y \sqsubseteq z \Rightarrow (y = \perp) \vee (y = z)}{p \ddagger x}\left\{\begin{array}{l}\text{provided } x{:}\tau \text{ is in the}\\ \text{variable context of } p\end{array}\right.$$

The above formalization of the logical calculus contains no big surprises. Most of it is similar to the LCF logic. Only in some rules the difference between two-valued and three-valued logic is apparent. Examples are [exmid], [impl], [TT_I], [TT_E], [eq_to_impl], [impl_to_eq], [DEF_ALLb], [DEF_EXb], [ALL_def], [EX_def], [strong_less], [strong_eq], [strong_DEF] and [fix_ind] and some of the axioms for the datatype Bool. In the rules [ext_less], [DEF_LAM] and [DEF_LAM$_n$] one can see that the space of continuous functions is lifted in SPECTRUM.

Theorem 1 *Correctness of the calculus*

If a sequent $H \blacktriangleright p$ over a given signature Σ and sort context $\mathcal{X}$ is derivable in the calculus formulated above then the sequent is also valid in every Σ-algebra where validity of sequents is defined according to definition 3.

Proof: Along the structure of the derivation for the sequent and the fact that all inference rules preserve validity.

4 Derived rules

In this section I will present some selected derived rules which all have been derived within the Isabelle system. I will not show all of the derived rules since I derived more than 180 inference rules. However all the rules presented below are given in the order they were proved to show their dependencies.

[refl] $\dfrac{}{\emptyset \blacktriangleright x = x}$

[sym] $\dfrac{H \blacktriangleright x = y}{H \blacktriangleright y = x}$

[trans] $\dfrac{H_1 \blacktriangleright x = y \quad H_2 \blacktriangleright y = z}{H_1, H_2 \blacktriangleright x = z}$

[cong] $\dfrac{H_1 \blacktriangleright s = t \quad H_2 \blacktriangleright x = y}{H_1, H_2 \blacktriangleright s(x) = t(y)}$

[ext_lemma1] $\dfrac{H_1 \blacktriangleright \delta(f) \quad H_2 \blacktriangleright \delta(g) \quad H_3 \blacktriangleright \forall^{\perp} x.f(x) = g(x)}{H_1, H_2, H_3 \blacktriangleright f = g}\left\{ x \notin FV(f) \cup FV(g)\right.$

[notnotE] $\dfrac{H \blacktriangleright \neg\neg p}{H \blacktriangleright p}$

[notnotI] $\dfrac{H \blacktriangleright p}{H \blacktriangleright \neg\neg p}$

[total_conj] $\dfrac{}{\emptyset \blacktriangleright \delta(x) \Rightarrow \delta(y) \Rightarrow \delta(x \wedge y)}$

[total_disj] $\dfrac{}{\emptyset \blacktriangleright \delta(x) \Rightarrow \delta(y) \Rightarrow \delta(x \vee y)}$

[total_impl] $\dfrac{}{\emptyset \blacktriangleright \delta(x) \Rightarrow \delta(y) \Rightarrow \delta(x \Rightarrow y)}$

[total_not] $\dfrac{}{\emptyset \blacktriangleright \delta(x) \Rightarrow \delta(\neg x)}$

[icontr] $\dfrac{H_1 \blacktriangleright p \quad H_2 \blacktriangleright \neg p}{H_1, H_2 \blacktriangleright \mathsf{false}}$

[defI] $\dfrac{H \blacktriangleright p}{H \blacktriangleright \delta(p)}$

[BOOLE] $\dfrac{H_1, x = \bot \blacktriangleright r \quad H_2, x = \mathsf{true} \blacktriangleright r \quad H_3, x = \mathsf{false} \blacktriangleright r}{H_1, H_2, H_3 \blacktriangleright r}$

[total_impl2] $\dfrac{H_1 \blacktriangleright \delta(p) \quad H_2, p \blacktriangleright \delta(q)}{H_1, H_2 \blacktriangleright \delta(p \Rightarrow q)}$

[DEF_BOOLE] $\dfrac{H \blacktriangleright \delta(t)}{H \blacktriangleright t = \mathsf{true} \vee t = \mathsf{false}}$

[UUE] $\dfrac{H \blacktriangleright \bot}{H \blacktriangleright p}$

[not_UUE] $\dfrac{H \blacktriangleright \neg\bot}{H \blacktriangleright p}$

[not_DEF_to_eq] $\dfrac{H_1 \blacktriangleright \neg\delta(t1) \quad H_2 \blacktriangleright \neg\delta(t2)}{H_1, H_2 \blacktriangleright t1 = t2}$

[less_UU_impl_eq_UU] $\dfrac{H \blacktriangleright x \sqsubseteq \bot}{H \blacktriangleright x = \bot}$

[ccontr] $\dfrac{H_1 \blacktriangleright \delta(p) \quad H_2, \neg p \blacktriangleright \mathsf{false}}{H_1, H_2 \blacktriangleright p}$

[EXbE_lemma2] $\dfrac{H \blacktriangleright \exists^{\perp} x.p}{H \blacktriangleright \exists^{\perp} x.p = \mathsf{true}}$

[EXbE2] $\dfrac{H_1 \blacktriangleright \exists^{\perp} x.p \quad H_2, p \blacktriangleright q}{H_1, H_2 \blacktriangleright q}$ provided $x \notin (FV(H_1) \cup FV(H_2) \cup FV(q))$

[EXbE_lemma3] $\dfrac{H \blacktriangleright \exists^{\perp} x.p = \mathsf{true}}{H \blacktriangleright \exists^{\perp} x.p}$

[ALLbE_lemma1] $\dfrac{H \blacktriangleright \forall^{\perp} x.p}{H \blacktriangleright \forall^{\perp} x.p = \mathsf{true}}$

[ALLbE_lemma2] $\dfrac{H \blacktriangleright \forall^{\perp} x.p = \mathsf{true}}{H \blacktriangleright \forall^{\perp} x.p}$

[conjE] $\dfrac{H_1 \blacktriangleright p \wedge q \quad H_2, p, q \blacktriangleright r}{H_1, H_2 \blacktriangleright r}$

[contrapos] $\dfrac{}{\emptyset \blacktriangleright (p \Rightarrow q) = (\neg q \Rightarrow \neg p)}$

[swap] $\dfrac{H_1 \blacktriangleright \delta(p) \quad H_2 \blacktriangleright \neg q \quad H_3, p \blacktriangleright q}{H_1, H_2, H_3 \blacktriangleright \neg p}$

[total_ALLb_lemma5] $\dfrac{H \blacktriangleright \forall^{\perp} x.\delta(p)}{H \blacktriangleright \delta(\forall^{\perp} x.p)}$

[total_EXb_lemma5] $\dfrac{H \blacktriangleright \forall^{\perp} x.\delta(p)}{H \blacktriangleright \delta(\exists^{\perp} x.p)}$

[not_ALLb_to_EXb_not] $\dfrac{}{\emptyset \blacktriangleright (\neg(\forall^{\perp} x.p)) = (\exists^{\perp} x.\neg p)}$

[not_EXb_to_ALLb_not] $\dfrac{}{\emptyset \blacktriangleright (\neg(\exists^{\perp} x.p)) = (\forall^{\perp} x.\neg p)}$

[DEF_DEF_impl_q_I] $\dfrac{H, \delta(x) \blacktriangleright \delta(q)}{H \blacktriangleright \delta(\delta(x) \Rightarrow q)}$ provided $x \notin FV(H)$

[ALLb_cong] $\dfrac{H \blacktriangleright \forall^{\perp} x.p = q}{H \blacktriangleright (\forall^{\perp} x.p) = (\forall^{\perp} x.q)}$

$$[\text{EXb\_cong}]\ \frac{H \blacktriangleright \forall^{\perp} x.p = q}{H \blacktriangleright (\exists^{\perp} x.p) = (\exists^{\perp} x.q)}$$

$$[\text{LAM\_cong\_lemma1}]\ \frac{H \blacktriangleright \forall^{\perp} x.t1 = t2}{H \blacktriangleright (\lambda x.t1) = (\lambda x.t2)}$$

The following rules were only proved for $2 \leq n \leq 5$ in Isabelle.

$$[\text{minimal\_prod}_n]\ \frac{}{\emptyset \blacktriangleright \langle \perp, \ldots, \perp \rangle \sqsubseteq \perp}$$

$$[\text{prod}_n\text{E}]\ \frac{H, p = \langle x_1, \ldots, x_n \rangle \blacktriangleright r}{H \blacktriangleright r} \left\{ \text{provided } x_1, \ldots, x_n \notin FV(H) \cup FV(r) \right.$$

$$[\text{invert\_prod}_n]\ \frac{H \blacktriangleright \langle x_1, \ldots, x_n \rangle \sqsubseteq \langle y_1, \ldots, y_n \rangle}{H \blacktriangleright x_1 \sqsubseteq y_1 \wedge \ \ldots \ \wedge x_n \sqsubseteq y_n}$$

$$[\text{inject\_prod}_n]\ \frac{H \blacktriangleright \langle x_1, \ldots, x_n \rangle = \langle y_1, \ldots, y_n \rangle}{H \blacktriangleright x_1 = y_1 \wedge \ \ldots \ \wedge x_n = y_n}$$

$$[\text{defined\_tuple}_n]\ \frac{H \blacktriangleright \delta(x_1) \vee \ \ldots \ \vee \delta(x_n)}{H \blacktriangleright \delta(\langle x_1, \ldots, x_n \rangle)}$$

$$[\text{LAM}_n\text{\_cong\_lemma1}]\ \frac{H \blacktriangleright \forall^{\perp} x_1 \ldots x_n.t1 = t2}{H \blacktriangleright (\lambda \langle x_1, \ldots, x_n \rangle.t1) = (\lambda \langle x_1, \ldots, x_n \rangle.t2)}$$

5 Conclusion

In this conclusion I would like to make some remarks about the logic of SPECTRUM. In the design of the language SPECTRUM [BFG+93a, BFG+93b] we tried to merge the level of boolean terms and that of formulae. This has the advantage that one needs only to introduce the logical connectives once and can use them both for boolean terms and formulae. Furthermore the specifications seem to have a nicer form and our feedback from writing specifications was positive.

However, the consequence of this identification of terms and formulae was a three-valued logic. Three-valued logics are frequently used in specification languages. One example is the logic LPF [Che86] the usability of which is discussed in [CJ90] and Olaf Owe gives a thorough discussion of various three-valued logics in [Owe91].

The three-valued logic of SPECTRUM that is presented in this paper was formalized in the Isabelle system. We did some case studies with this formalization [Pus94] and therefore gained some experience with this kind of three-valued logic. Also the derivation of more than 180 derived rules gave insight into the nature of SPECTRUM's three-valued logic. As a final judgment I would say that in principle it is possible to do proofs with this logic and especially with its formalization in Isabelle. But I also would like to add some critical remarks:

1. Reasoning in three-valued logic seems to be quite artificial in some cases and there is often a surprise when a theorem that holds in two-valued logic is not valid in the three-valued case. Moreover, I think that the definedness logic of LCF is messy enough at the level of terms and it is not a profit that the reasoning about definedness is propagated to the level of formulae.

2. The degree of automation is very low in the Isabelle formalization and therefore there is not much fun in doing a bigger case study. The reason for this poor theorem proving support are subgoals about the definedness of terms that frequently pop up during proofs. Although we have implemented some (very primitive) tactics that automatically prove the definedness of a term, this did not improve the degree of automation very much. It seems that the generic proof tools of Isabelle, especially the classical prover which we could not instantiate for SPECTRUM, are better suited for two-valued logics. I think that a satisfactory degree of automation can only be achieved in a theorem proving environment that is tuned especially for three-valued logics.

As a consequence of these results I developed an enhanced variant of the SPECTRUM logic. This logic is called HOLCF and is a higher-order version of LCF. It is based on Isabelle's formalization of HOL [Pau93b] and is therefore a two-valued logic again. Despite the three-valued logic and the identification of formulae and terms of the datatype Bool it incorporates all the features of the SPECTRUM logic. Moreover, since it allows full higher-order logic its expressiveness is far beyond that of LCF or SPECTRUM. A thorough description of the logic HOLCF is given in [Reg94]. The Isabelle formalization of HOLCF is part of the Isabelle distribution.

References

[BFG+93a] Manfred Broy, Christian Facchi, Radu Grosu, Rudi Hettler, Heinrich Hussmann, Dieter Nazareth, Franz Regensburger, Oscar Slotosch, and Ketil Stølen. The Requirement and Design Secification Language SPECTRUM. An Informal Introduction. Version 1.0. Part i. Technical Report TUM-I9311, Technische Universität München. Institut für Informatik, Fakultät für Informatik, TUM, 80290 München, Germany, May 1993.

[BFG+93b] Manfred Broy, Christian Facchi, Radu Grosu, Rudi Hettler, Heinrich Hussmann, Dieter Nazareth, Franz Regensburger, Oscar Slotosch, and Ketil Stølen. The Requirement and Design Secification Language SPECTRUM. An Informal Introduction. Version 1.0. Part ii. Technical Report TUM-I9312, Technische Universität München. Institut für Informatik, Fakultät für Informatik, TUM, 80290 München, Germany, May 1993.

[Che86] J.H. Cheng. *A Logic for Partial Functions.* PhD thesis, Departement of Computer Science University of Manchester, 1986. Technical Report Series UMCS-86-7-1.

[CJ90] J.H. Cheng and C.B. Jones. On the Usability of Logics which Handle Partial Functions. *Technical Report Series UMCS-90-3-1, Departement of Computer Science University of Manchester*, 1990.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935.

[GM93] M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[GR94] Radu Grosu and Franz Regensburger. The Logical Framework of SPECTRUM. Technical Report TUM-I9402, Institut für Informatik, Technische Universität München, 1994.

[Nip91] Tobias Nipkow. Order-Sorted Polymorphism in Isabelle. In G. Huet, G. Plotkin, and C. Jones, editors, *Proc. 2nd Workshop on Logical Frameworks*, pages 307–321, 1991.

[Owe91] O. Owe. Partial Logics Reconsidered: A Conservative Approach. Research Report 155, Departement of Informatics, University of Oslo, June 1991.

[Pau87] L.C. Paulson. *Logic and Computation, Interactive Proof with Cambridge LCF*, volume 2 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1987.

[Pau93a] L.C. Paulson. The Isabelle Reference Manual. Technical Report 283, University of Cambridge, Computer Laboratory, 1993.

[Pau93b] L.C. Paulson. Isabelle's Object Logics. Technical Report 286, University of Cambridge, Computer Laboratory, 1993.

[Pus94] Cornelia Pusch. Verifikation einer Entwicklung von AVL-Bäumen in Isabelle. Diplomarbeit, Technische Universität München, 1994.

[Reg94] Franz Regensburger. *HOLCF: Eine konservative Erweiterung von HOL um LCF*. PhD thesis, Technische Universität München, 1994.

[Sch86] D.A. Schmidt. *Denotational Semantics*. Allan and Bacon, 1986.