Anna Zalewska
University of Białystok

PROBLEMS WITH ω–RULES IN MECHANIZATION OF REASONING. A COMPARISON OF TWO SYSTEMS

1. Motivation

Gödel's First Incompleteness Theorem (Gödel, 1931) showed that in any consistent formal system powerful enough to do a certain sort of arithmetic there will be a true sentence that the system cannot prove. The theorem indicated a certain gap in Hilbert's programme (Hilbert, 1901): truth cannot be achieved by provability; it can be only approximated by syntactic means. How to extend the Hilbert's finitistic point of view and to overcome incompleteness of formal axiomatic systems? Hilbert in 1930 proposed to admit a new inference rule to be able to realize his program (it was a some kind of informal ω–rule). Also Turing in 1939 (Turing, 1939) pursued Church's idea for his Ph. D. thesis, looking to ordinal logic as a way to "escape" Gödel's incompleteness theorem. Now several approaches to the problem in question are known among them admitting the ω–rule, adding new axioms or adding (partial) notions of truth.

In logics of programs (such as dynamic logic (Harel, 1979), Hoare–like logics (Hoare, 1969) or algorithmic logic (Mirkowska, Salwicki, 1987; Mirkowska, 1971, 1981)) by reason of complexity there also do not exist formal finitistic proof systems that are complete in the classical sense. There are two fundamental approaches to the construction of these logics: to limit a class of considered interpretations or to admit some ω–rules. Algorithmic logic is a result of the second approach.

It is obvious that a formal verification of program correctness is a very important task. Consequences of possible errors in programs may prove expensive and sometimes irreversible, particularly nowadays when computers have become indispensable in all domains of our life. But the formal proofs of program properties in algorithmic logic (and also in dynamic logic or Hoare–like logics) are often nontrivial and with a large number of tedious technical details. For that reason no wonder that there is a growing need for mechanization of the proof process. The main obstacle in the mechanization of reasoning in algorithmic logic are, first of all, the ω–rules which are not implemented.

In the present paper two formal systems (S_{MIND} and S_{LOOP}) for the logic are compared. Each of them is suitable for computer realization but tries to overcome the problem of ω–rules in the different way: the first of them replaces ω–rules by metainduction; the second of them uses the notion of proof as a finite tree with redundant nodes, i.e. looping nodes satisfying some conditions.

2. Introduction

Both the S_{MIND} system and the S_{LOOP} system are finite cut–free Gentzen–type axiomatizations for propositional algorithmic logic (PAL).

The syntax of PAL is based upon two sets of symbols:

V_0 – an enumerable set of propositional variables,

V_p – an enumerable set of program variables.

From V_0 we construct the set of open formulas F_0 as usual propositional formulas (i.e. $V_0 \in F_0$) and if α, $\beta \in F_0$ then $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $\neg\alpha \in F_0$ are propositional formulas.

Given sets F_0 and V_p the set Pr of *programs* is generated by the following grammar:

$$Pr ::= V_p \mid (Pr; Pr) \mid \textbf{if } F_0 \textbf{ then } Pr \textbf{ else } Pr \textbf{ fi} \mid \textbf{while } F_0 \textbf{ do } Pr \textbf{ od}.$$

The set of all *formulas* F is defined as follows

$$F ::= F_0 \mid F \vee F \mid F \wedge F \mid \neg F \mid MF.$$

Let B_0 be two–element Boolean algebra. By a *semantic structure* $\mathfrak{M}$ we shall understand a triple $\langle S, \mathcal{I}, w\rangle$ where S is a nonempty set of states, $\mathcal{I} : V_p \to 2^{S\times S}$ is an interpretation of the program variables (where $\mathcal{I}(Id) = \{(s,s) : s \in S\}$) and $w : S \to B_0^{V_0}$ is a function assigning to every state a valuation of propositional variables. For every program variable K and three states s, s_1, s_2 the following condition (condition of deterministic programs) is satisfied:

$$\text{if } (s, s_1) \in \mathcal{I}(K) \text{ and } (s, s_2) \in \mathcal{I}(K), \text{ then } s_1 = s_2.$$

For a given structure $\mathfrak{M}$ and a given state $s \in S$ the Boolean value of the formula α is denoted by $\alpha_{\mathfrak{A}}(s)$ and is defined for classical connectives as follows:

$$p_{\mathfrak{A}}(s) = w(s)(p) \quad (p \in V_o), \qquad (\neg\alpha)_{\mathfrak{A}}(s) = -\alpha_{\mathfrak{A}}(s),$$
$$(\alpha \vee \beta)_{\mathfrak{A}}(s) = \alpha_{\mathfrak{A}}(s) \cup \beta_{\mathfrak{A}}(s), \qquad (\alpha \wedge \beta)_{\mathfrak{A}}(s) = \alpha_{\mathfrak{A}}(s) \cap \beta_{\mathfrak{A}}(s),$$

Let us denote by $K_{\mathfrak{A}}$ a relation which is assigned to a program variable K by interpretation $\mathcal{I}$ in the semantic structure $\mathfrak{M}=\langle S, \mathcal{I}, w\rangle$. Let s be a state and M, M' be arbitrary programs then

$Id_{\mathfrak{A}}(s) = s,$

$K_{\mathfrak{A}}(s) = \{s' : (s, s') \in K_{\mathfrak{A}}\}$ and $K_{\mathfrak{A}}(s)$ is at most an one–element set,

$$(M; M')_{\mathfrak{A}}(s) = \bigcup_{s' \in M_{\mathfrak{M}}(s)} M'_{\mathfrak{A}}(s'),$$

$$\textbf{if } \gamma \textbf{ then } M \textbf{ else } M' \textbf{ fi}_{\mathfrak{A}}(s) = \begin{cases} M_{\mathfrak{A}}(s) & \text{if } (\gamma)_{\mathfrak{A}}(s) = 1, \\ M'_{\mathfrak{A}}(s) & \text{if } (\neg\gamma)_{\mathfrak{A}}(s) = 1, \end{cases}$$

$$\textbf{while } \gamma \textbf{ do } M \textbf{ od}_{\mathfrak{A}}(s) = \bigcup_{i \in N} \{s' \in (\textbf{if } \gamma \textbf{ then } M \textbf{ fi})^i_{\mathfrak{A}}(s) : (\neg\gamma)_{\mathfrak{A}}(s') = 1\}.$$

For an arbitrary state s in the structure $\mathfrak{M}$ we assume that

$$(M\alpha)_{\mathfrak{A}}(s) = 1 \text{ iff there exists the state } s' \in M_{\mathfrak{A}}(s) \text{ such that } (\alpha)_{\mathfrak{A}}(s') = 1.$$

We shall write $\mathfrak{M}$, $s \models \alpha$ in order to denote that $\alpha_{\mathfrak{A}}(s) = 1$. The formula α is *valid in the structure* $\mathfrak{M}$ ($\mathfrak{M} \models \alpha$) iff for every state $s \in S$ holds $\mathfrak{M}$, $s \models \alpha$. In the rest of the paper $(\alpha \Rightarrow \beta)$ is an abbreviation of the formula $(\neg\alpha \vee \beta)$ and $(\alpha \Leftrightarrow \beta)$ is an abbreviation of the formula $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$. Let us denote by **true** the formula $(p \vee \neg p)$ and by **false** the formula $(p \wedge \neg p)$, for a fixed propositional variable p.

3. The S_{MIND} system

The S_{MIND} system correspond to earlier research on prover for algorithmic logic of Zalewska (1996, 2001). In this section we would like to recall some basic notions connected with the system.

The main idea of the S_{MIND} system is as follows: replace ω-rules such as the below one

$$\frac{\{\Gamma, \neg\textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha, \Delta\}}{\{\Gamma, \Delta, \neg(\textbf{if } \gamma \textbf{ then } M \textbf{ fi})^i(\neg\gamma \wedge \alpha)\}_{i \in N}}$$

by *metainduction*, i.e. by the rule as follows

$$\frac{\langle \{\Gamma, \neg \textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha, \Delta\},\ \mathcal{A}\rangle}{\langle \{\Gamma, \Delta, \neg\delta\},\ \mathcal{A}\ \rangle\ ;\ \langle \{\Gamma, \Delta, \neg\ \mathrm{IF}(\mathrm{IF}^j\delta)\},\ \mathcal{A} \cup \{\{\Gamma, \Delta, \neg \mathrm{IF}^j\delta\}\}\rangle}$$

where $\delta = \neg\gamma \wedge \alpha$, IF $=$ **if** γ **then** M **fi**, j is a parameter of natural type.

It allows to prove only that the proof exists instead of carrying out full proof for a given formula. The notion of metainduction is formalized in this way that the conclusion and premises of rules are presented as ordered pair of the form $\langle \Pi, \mathcal{A}\rangle$ where Π denotes the main sequent and $\mathcal{A}$ stands for set (maybe empty) of sequents that are called *metainduction assumptions.* The notion of validity of the main sequent Π of the ordered pair $\langle \Pi, \mathcal{A}\rangle$ with respect to $\mathcal{A}$ is defined in the following way: Π *is valid assuming that each sequent from the set* $\mathcal{A}$ *is also valid.* In our calculus ω-rules are replaced by *metainduction rules.* In consequence the standard notion of sequent and process of inferences is modified. The main sequent Π of the ordered pair $\langle \Pi, \mathcal{A}\rangle$ is said to be

- *indecomposable* iff no rule can be applied to it;
- *fundamental* iff the formulas α and $\neg\alpha$ belongs to the sequent Π;
- $\mathcal{A}$*–provable* iff there exists a sequent $\Sigma \in \mathcal{A}$ such that $\Sigma \subseteq \Pi$
  (we will say sometimes that Π is $\mathcal{A}$*–provable with respect to the sequent* Σ);
- A**–provable* iff there exists a sequent $\Sigma \in \mathcal{A}$ for which at least one formula $\beta \in \Sigma$ is with negation and there exists a program M such that

  $$\forall_{\alpha\in\Sigma}\exists_{\beta\in\Pi}(\beta = \alpha_M)$$

  where $\alpha_M = \pm M\alpha'$ if $\alpha = \pm\alpha'$ and $\pm \in \{\neg, \varepsilon\}$
  (we will say sometimes that Π is $\mathcal{A}^*$*–provable with respect to the sequent* Σ *and the program* M);
- *terminal* iff Π is indecomposable but Π is not fundamental and neither $\mathcal{A}$–provable nor $\mathcal{A}^*$–provable.

A *proof* of the sequent Π is a diagram (diagram is a decomposition tree obtaining by application of decomposition rules to the input formula) of the sequent such that all paths of the diagram are finite and each its leaf is labelled by the ordered pair $\langle \Pi, \mathcal{A}\rangle$ where Π is fundamental or $\mathcal{A}$-provable or $\mathcal{A}^*$-provable.

4. The S_{LOOP} system

The S_{LOOP} system correspond to earlier researches on the finite Gentzen–like axiomatization for PAL of Chlebus (1982) and Walukiewicz (1990). In this section we would like to present the system for some deterministic version of PAL.

The main rules of the S_{LOOP} system (where upper sequents of rules will be called *assumptions*, lower will be called *conclusion* and $\mp \in \{\neg, \epsilon\}$) are as follows:

(1)

$$\frac{\{\Gamma, \neg\neg\alpha, \Delta\}}{\{\Gamma, \alpha, \Delta\}}$$

(2)

$$\frac{\{\Gamma, \mp Id\alpha, \Delta\}}{\{\Gamma, \mp\alpha, \Delta\}}$$

(3)

$$\frac{\{\Gamma, \mp(\alpha\Box\beta), \Delta\}}{\{\Gamma, \mp\alpha, \Delta\}; \{\Gamma, \mp\beta, \Delta\}}$$

where $(\mp, \Box) \in (\epsilon, \wedge), (\neg, \vee)\}$

(4)

$$\frac{\{\Gamma, \mp(\alpha\Box\beta), \Delta\}}{\{\Gamma, \mp\alpha, \mp\beta, \Delta\}}$$

where $(\mp, \Box) \in \{(\epsilon, \vee)(\neg, \wedge)\}$

(5)

$$\frac{\{\Gamma, \mp(M'; M'')\alpha, \Delta\}}{\{\Gamma, \mp M'(M''\alpha), \Delta\}}$$

(6)

$$\frac{\{\Gamma, \mp\textbf{if } \gamma \textbf{ then } \text{M'} \textbf{ else } M'' \textbf{ fi}\alpha, \Delta\}}{\{\Gamma, \mp((\gamma \wedge M'\alpha) \vee (\neg\gamma \wedge M''\alpha)), \Delta\}}$$

(7)

$$\frac{\{\Gamma, \neg\textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha, \Delta\}}{\{\Gamma, \neg\alpha, \gamma, \Delta\};\ \{\Gamma, \neg\gamma, \neg M(\textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha), \Delta\}}$$

(8)

$$\frac{\{\Gamma, \textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha, \Delta\}}{\{\Gamma, \alpha, \gamma, \Delta\};\ \{\Gamma, \neg\gamma, M(\textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha), \Delta\}}$$

$$\frac{\{\Gamma, \neg M\alpha, \Delta\}}{\{\{\beta : M\beta \in \Gamma \textit{ or } \neg M\beta \in \Gamma\}, \neg\alpha,\ \{\beta : M\beta \in \Delta \textit{ or } \neg M\beta \in \Delta\}\}} \quad (9)$$

Given a sequence of sequents $\{\Gamma_i, \Delta_i\}_{i \in I}$ such that $\{\Gamma_{i+1}, \Delta_{i+1}\}$ is an assumption and $\{\Gamma_i, \Delta_i\}$ a conclusion in some rule of the S_{LOOP} system, we define:

– family of *trace relations* $R_{i,i+j}(i+j \in I, i, j > 0)$ in the following way:
  1. if formula $\alpha \in \Gamma_i$ is not reduced by the rule applied to $\{\Gamma_i, \Delta_i\}$ then $(\alpha, \alpha) \in R_{i,i+1}$;
  2. if formula $K\alpha \in \Gamma_i$ and a rule applied to $\{\Gamma_i, \Delta_i\}$ reduced it to the formula $M\beta$ then $(K\alpha, M\beta) \in R_{i,i+1}$;
  3. otherwise $R_{i,i+j} = R_{i,i+j-1} \circ R_{j-1,j}$ $(j > 1)$;

– *trace* of any formula $\alpha \in \Gamma_i$ to be a sequence $(\alpha_j)_{j \in J}$ such that $\alpha_j \in R_{i+j}$;

– *loop node* as such element $i \in I$ that there exists $j < i, \{\Gamma_i, \Delta_i\} = = \{\Gamma_j, \Delta_j\}$ and $R_{j,i} = \emptyset$;

– *redundant node* as such element $i \in I$ that there exists $j_1 < j_2 < i$, such that
  1. $\{\Gamma_i, \Delta_i\} = \{\Gamma_{j_1}, \Delta_{j_1}\} = \{\Gamma_{j_2}, \Delta_{j_2}\}$;
  2. $R_{j_1,i} = R_{j_1,j_2} \neq \emptyset$;
  3. $\{\{\Gamma_k, \Delta_k\} : j_1 < k < i\} \subseteq \{\{\Gamma_k, \Delta_k\} : k < j_1\}$;
  4. there is no loop node between j_1 and i and between j_1 and j_2.

A *proof* of the sequent Π is a diagram (diagram is a decomposition tree obtaining by application of decomposition rules to the input formula) of the sequent such that all paths of the diagram are finite and each its leaf is labelled by the sequent Π_n where n is redundant node or Π_n is axiom (i.e. $\{\alpha, \neg\alpha\} \subseteq \Pi_n$).

EXAMPLE. Let us consider the following formula:

$$\{\textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha \ \rightarrow\ \textbf{while } \gamma \textbf{ do } M \textbf{ od}(\neg\gamma)\}.$$

In the proof process, after the application of rules for logical connectives: $\rightarrow$, we obtain the following sequent:

$$(*) \quad \{\neg\textbf{while } \gamma \textbf{ do } M \textbf{ od}\alpha,\ \textbf{while } \gamma \textbf{ do } M \textbf{ od}(\neg\gamma)\}.$$

After the application of the rule (7) to the sequent we obtain two new sequents:

(1) $\{\neg\gamma, \neg M(\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha),\ \textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}(\neg\gamma)\}$

(2) $\{\neg\alpha,\ \gamma,\ \textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}(\neg\gamma)\}$.

We can apply the rule (8) to the sequent (2). In this way we obtain as follows:

(2.1) $\{\neg\alpha,\ \neg\gamma,\ \gamma,\ \textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}(\neg\gamma)\}$

(2.2) $\{\neg\alpha, \neg\gamma,\ \gamma,\ \}$

Both of the above sequents are axioms. Now we can apply the rule (7) to the sequent (1). In this way we have two sequents:

(1.1) $\{\neg\gamma, \neg M(\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha),\ \gamma,\ \neg\gamma\}$

(1.2) $\{\neg\gamma, \neg M(\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha),\ M(\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}(\neg\gamma))\}$

The sequent (1.1) is an axiom. From (1.2) (after the application of the rule (9)) we obtain the following sequent:

(1.2.1) $\{\neg\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha,\ \textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}(\neg\gamma)\}$

The sequent (1.2.1) is equal to the sequent (*). It is not a redundant sequent but repeating the proof for this sequent we obtain a redundant one.

5. Comparison of the systems

Both systems are sound, complete and decidable. In the section we prove derivability of the S_{LOOP} rules in the S_{MIND} system. Next we shall discuss the problem of the application of the systems to the first–order algorithmic logic.

Theorem. For every rule r if r is the S_{LOOP} rule then r is derivable in the S_{MIND} system.

Proof. The rule (1)–(6) of the S_{LOOP} system are the same as the proper rules of the S_{MIND} system.

The rule (7). In order to derive the rule (7) in the S_{MIND} system we have to prove the main sequent of the following ordered pair

$$\langle\{\neg\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha, \},\quad \mathcal{A} = \{\{\neg\alpha, \gamma, \},\ \{\neg\gamma, \neg M(\textbf{while}\ \gamma\ \textbf{do}\ M\ \textbf{od}\alpha)\}\}\rangle.$$

After the application of the rule (1) to the main sequent and to the elements of the set $\mathcal{A}$ we obtain the following sequent:

$$\langle\{\neg\bigcup\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha),\},$$
$$\mathcal{A}=\{\{\neg\alpha,\gamma,\},\{\neg\gamma,\neg M\bigcup\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha)\}\}\rangle.$$

Now we can apply the metainduction rule (3) to the main sequent. We have two following sequents:

(1) $\langle\{\neg(\neg\gamma\wedge\alpha),\},$
$\mathcal{A}=\{\{\neg\alpha,\gamma,\},\ \{\neg\gamma,\neg M\bigcup\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha)\},\}\rangle.$

(2) $\langle\{\neg\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}(IF^i(\neg\gamma\wedge\alpha)),\},\ \mathcal{A}'=\mathcal{A}\cup\{\ \{\neg IF^i(\neg\gamma\wedge\alpha)\}\}\rangle.$

The sequent (1), after the application of rules for logical connectives, is $\mathcal{A}$–provable with respect to the assumption $\{\neg\alpha,\gamma\}$:

$$\langle\{\gamma,\neg\alpha),\},\ \mathcal{A}=\{\{\neg\alpha,\gamma,\},\ \{\neg\gamma,\neg M\bigcup\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha)\}\}\rangle.$$

From the sequent (2), after the application of the rule for the program connective **if** – **then** – **fi**, rules for logical connectives and the rule (4), we obtain two sequents:

(2.1) $\langle\{\neg\gamma\ \neg M(IF^i(\neg\gamma\wedge\alpha)),\},$
$\mathcal{A}'=\mathcal{A}\cup\{\{\neg IF^i(\neg\gamma\wedge\alpha)\},\{\neg\gamma,\neg M(IF^i(\neg\gamma\wedge\alpha))\}\}\rangle.$

(2.2) $\langle\{\gamma\ ,\neg IF^i(\neg\gamma\wedge\alpha)),\},$
$\mathcal{A}'=\mathcal{A}\cup\{\{\neg IF^i(\neg\gamma\wedge\alpha)\},\{\neg\gamma,\neg M(IF^i(\neg\gamma\wedge\alpha))\}\}\rangle.$

The first of them is $\mathcal{A}'$–provable with respect to the sequent $\{\neg\gamma,\neg M(IF^i(\neg\gamma\wedge\alpha))\}$. The second of them is $\mathcal{A}'$–provable with respect to the sequent $\{\neg IF^i(\neg\gamma\wedge\alpha)\}$.

The rule (8). In order to derive the rule (8) in the S_{MIND} system we have to prove the main sequent of the following ordered pair

$$\langle\{\mathbf{while}\ \gamma\ \mathbf{do}\ M\ \mathbf{od}\alpha,\},$$
$$\mathcal{A}=\{\{\alpha,\gamma\},\ \{\neg\gamma,M(\mathbf{while}\ \gamma\ \mathbf{do}\ M\ \mathbf{od}\alpha)\}\}\rangle.$$

After the application of the rule (1) to the main sequent and to the elements of the set $\mathcal{A}$ we obtain the following sequent:

$$\langle\{\bigcup\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha),\},$$
$$\mathcal{A}=\{\{\alpha,\gamma\},\ \{\neg\gamma,M\bigcup\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{od}(\neg\gamma\wedge\alpha)\}\}\rangle.$$

Now we can apply the rule (2) to the main sequent. We have the following sequent:

$$\langle\{\neg\gamma\wedge\alpha,\mathbf{if}\ \gamma\ \mathbf{then}\ M\ \mathbf{fi}\bigcup\mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha))\},\ \mathcal{A}\}\rangle.$$

From the sequent, after the application of the rule for the program connective **if** – **then** – **fi** and rules for logical connectives, we obtain as follows:

(1) $\langle\{\neg\gamma,\gamma,\neg\gamma\wedge\bigcup\mathbf{then}\ M\ \mathbf{fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle,$

(2) $\langle\{\neg\gamma, M \bigcup \textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha), \neg\gamma\wedge\bigcup\textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle$,
(3) $\langle\{\alpha, \gamma, \neg\gamma\wedge\bigcup\textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle$,
(4) $\langle\{\alpha, M \bigcup \textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha), \neg\gamma\wedge\bigcup\textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle$.

The sequent (1) is fundamental. The sequent (2) is $\mathcal{A}$–provable with respect to the sequent $\{\neg\gamma, M\bigcup\textbf{if } \gamma \textbf{ then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\}$. The sequent (3) is $\mathcal{A}$–provable with respect to the sequent $\{\alpha,\gamma\}$. Let us observe that the sequent (4) is equivalent to the following sequent:

$\langle\{(\gamma\wedge\neg\gamma), \alpha, M \bigcup \textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha), \neg\gamma\wedge\bigcup\textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle$.

After the application of the rule for logical connective $\wedge$ we obtain two sequent:

$\langle\{\gamma, \alpha, M \bigcup \textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha), \neg\gamma\wedge\bigcup\textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle$,
$\langle\{\neg\gamma, \alpha, M \bigcup \textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha), \neg\gamma\wedge\bigcup\textbf{then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\},\ \mathcal{A}\}\rangle$.

The first of them is $\mathcal{A}$–provable with respect to the sequent $\{\alpha,\gamma\}$. The second of them is $\mathcal{A}$–provable with respect to the sequent $\{\neg\gamma, M\bigcup\textbf{if } \gamma \textbf{ then } M \textbf{ fi}(\neg\gamma\wedge\alpha)\}$.

The rule (9). In order to derive the rule (8) in the S_{MIND} system we have to prove the main sequent of the following ordered pair

$\langle\{\Gamma, \neg M\alpha, \Delta\},\ \mathcal{A} =$
$\{\{\{\beta : M\beta\in\Gamma \textit{ or } \neg M\beta\in\Gamma\}, \neg\alpha, \{\beta : M\beta\in\Delta \textit{ or } \neg M\beta\in\Delta\}\}\}\rangle$.

Let us observe that the main sequent is $\mathcal{A}^*$–provable with respect to assumption and the program M. □

Because

any valid formula of propositional algorithmic logic becomes a valid formula of the first-order algorithmic logic following a substitution of formulas for propositional variables and programs for programs variables

we can extend each of the systems presented in the paper to the first–order algorithmic logic (AL). Of course AL is not complete. We can only try to enlarge the class of provable formulas in a given extended system by some modifications and other new rules. In order to do this we have to base on some kinde of "open" system which allows to do such modifications and which allows to extend the basic system by new rules in natural way.

Let us consider the following AL formula:

$\langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),$
$\textbf{while } \neg P(x) \textbf{ do } x := f(x) \textbf{ od} P(x)\},\ \emptyset\rangle$.

The proof in the S_{MIND} system is presented below.

After the application of the rule (1) to the main sequent we obtain the following sequent:

$$\begin{aligned}&\langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\\ &\qquad\qquad\qquad \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x)\},\ \emptyset\rangle.\end{aligned}$$

Now we can apply the rule (2) to the main sequent. We obtain the following one:

$$\begin{aligned}&\langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x),\\ &\quad \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x)\},\ \emptyset\rangle.\end{aligned}$$

From the sequent, after the application of the rule for the program connective **if** – **then** – **fi** and rules for logical connectives, we have the following sequents (where $\delta = P(x) \wedge \bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x)$):

$$(1)\quad \langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x),\ \neg P(x),\ \delta\},\ \emptyset\rangle$$

$$\begin{aligned}(2)\quad &\langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x),\\ &\quad (x := f(x)) \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x), \delta\},\ \emptyset\rangle.\end{aligned}$$

The sequent (1) is fundamental. Now we can apply to the sequent (2) again the rule (2):

$$\begin{aligned}&\langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x),\ (x := f(x))P(x),\\ &\quad (x := f(x))\textbf{if } \neg P(x) \textbf{ then}\\ &\qquad\qquad x := f(x) \textbf{ fi} \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x), \delta\},\ \emptyset\rangle.\end{aligned}$$

This sequent is also fundamental (after the application of the rule (5)):

$$\begin{aligned}&\langle\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x),\ P(f(x)),\\ &\quad (x := f(x))\textbf{if } \neg P(x) \textbf{ then}\\ &\qquad\qquad x := f(x) \textbf{ fi} \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x), \delta\},\ \emptyset\rangle.\end{aligned}$$

Now we shall try to prove the same formula in the S_{LOOP} system.

After the application of the rule (8) we obtain the following two sequents:

$$\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x)\ \neg P(x)\},$$

$$\begin{aligned}&\{\neg(x := f(x))Q(x),\ \neg P(f(x)),\ P(x),\\ &\qquad\qquad (x := f(x) \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x)\}.\end{aligned}$$

The first sequent is an axiom. Now we can apply the rule (9) to the second sequent for $M = (x := f(x))$:

$$\{\neg Q(x), \textstyle\bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x)\}.$$

Let us observe that the formula $\neg P(f(x))$ has been removed from our sequent. Of course we can again apply the rule (8) obtaining in that way two sequents:

$\{\neg Q(x),\ P(x),\ \neg P(x)\}$,

$\{\neg Q(x), P(x),\ (x := f(x)) \bigcup \textbf{if } \neg P(x) \textbf{ then } x := f(x) \textbf{ fi} P(x)\}$.

The first sequent is fundamental but the second of them is not provable in the S_{LOOP} system.

Let us recapitulate the comparison of the systems. Let $\mathcal{C}_{LOOP}$ be the class of provable formulas in the S_{LOOP} system, $\mathcal{C}_{MIND}$ be the class of provable formulas in the S_{MIND} system, $\overline{\mathcal{C}_{LOOP}}$ be the class of provable formulas in the S_{LOOP} system extended to AL and $\overline{\mathcal{C}_{MIND}}$ be the class of provable formulas in the S_{MIND} system extended to AL. The following relations describe the dependences between these classes:

1. $\mathcal{C}_{LOOP} = \mathcal{C}_{MIND}$
2. $\overline{\mathcal{C}_{LOOP}} \subset \overline{\mathcal{C}_{MIND}}$.

References

Chlebus, B. (1982), *On the Decidability of Propositional Algorithmic Logic*, Zeitschr. f. math. Logik und Grundlagen d. Math., Bd 28, 247-261.

Gödel, K. (1931), *Über formal unentscheidbare Sätze der "Principia mathematica" und verwandter Systeme I.*, Monats. Math. Ph. 38, 173–198.

Harel D. (1979), *First Order Dynamic Logic*, Sppringer, LNCS 68.

Hilbert, D. (1901), *Mathematische Probleme*, Archiv der Mathematik und Physik 1, 44–63 and 213–237.

Hoare C.A.R. (1969), *An axiomatic basis for computer programming*, Comm. Assoc. Comput. Mach. 12, 576-580, 583.

Mirkowska G. (1971), *On Formalized Systems of Algorithmic Logic*, Bull. PAS 19, 421-428.

Mirkowska G. (1981), *PAL - Propositional Algorithmic Logic*, Fundamenta Informaticae 4, pp 675-757 (also: Logics of Programs (E.Engeler ed.) LNCS 125, Springer-Verlag, 12-22).

Mirkowska G., Salwicki A. (1987), *Algorithmic Logic*, D. Reidel Publishing Company, Dordrecht

Turing A.M. (1939), *Systems of logic based on ordinals*, P. Lond. Math. Soc. (2) 45, 161–228.

Walukiewicz, I. (1990), *Gentzen Type Axiomatization for Propositional Algorithmic Logic*, MFCS'90 (B.Rovan ed.), LNCS 452, Springr--Verlag, 499-508 (Full version in TCS 118 (1993), 67-79.

Zalewska, A. (1996), *Interactive Prover for Programs Properties Expressed in Algorithmic Logic*, Doctoral Dissertation, Warsaw University.

Zalewska, A. (2001), *Program Verification with Algorithmic Logic*, Philomath, Warsaw University.

Appendix

Decomposition rules for S_{MIND} system

We present only rules which are used (in the proof) in the present paper and are not the same as the rules of the S_{LOOP} system.

Let Γ denotes a set of indecomposable formulas; Δ is an arbitrary set of formulas; α, β are arbitrary formulas; γ denotes a propositional formula; j is a parameter of natural type; M, M', M'' denote arbitrary programs; $\circ \in \{-, +\}$; $\mp \in \{\neg, \epsilon\}$; $\Box \in \{\wedge, \vee\}$; $\mathrm{Q} \in \{\bigcup, \bigcap\}$; *pref*$^\circ$ is a sequence of simple programs.

(1)

$$\frac{\{\Gamma, \mathit{pref}^\circ \mp \textbf{while } \gamma \textbf{ do } M \textbf{ od} \alpha, \Delta\}}{\{\Gamma, \mathit{pref}^\circ \mp \bigcup \textbf{ if } \gamma \textbf{ then } M \textbf{ fi } (\neg\gamma \wedge \alpha), \Delta\}}$$

(2)

$$\frac{\langle \{\Gamma, \mathit{pref}^\circ \mp \mathrm{Q} M \alpha, \Delta\}, \mathcal{A}\rangle}{\langle \{\Gamma, \mathit{pref}^\circ \mp \alpha, \Delta, \mathit{pref}^\circ \mp M \mathrm{Q} M \alpha\}, \mathcal{A}\rangle}$$

where $(\circ, \mp, \mathrm{Q}) \in \{(+, \epsilon, \bigcup), (-, \epsilon, \bigcap), (-, \neg, \bigcup), (+, \neg, \bigcap)\}$

(3)

$$\frac{\langle \{\Gamma, \mathit{pref}^\circ \mp \mathrm{Q} M \alpha, \Delta\}, \mathcal{A}\rangle}{\begin{array}{c}\langle \{\Gamma, \mathit{pref}^\circ \mp \alpha, \Delta\}, \mathcal{A}\text{>}; \text{<}\{\Gamma, \Delta, \mathit{pref}^\circ \mp M(IdM^j \alpha)\}, \\ \mathcal{A} \cup \{\{\Gamma, \Delta, \mathit{pref}^\circ \mp IdM^j \alpha\}\}\rangle\end{array}}$$

where $(\circ, \mp, \mathrm{Q}) \in \{(-, \epsilon, \bigcup), (+, \epsilon, \bigcap), (+, \neg, \bigcup), (-, \neg, \bigcap)\}$

(4)

$$\frac{\langle \Pi, \mathcal{A} = \mathcal{A}' \cup \{\{\Gamma', \mathit{pref}^{\circ} \mp \mathrm{Q} M \alpha, \Delta'\}\}\rangle}{\langle \Pi, \mathcal{A} \cup \{\{\Gamma', \mathit{pref}^{\circ} \mp \alpha, \Delta'\}, \{\Gamma', \mathit{pref}^{\circ} \mp IdM^{j} \alpha, \Delta'\}\}\rangle}$$

where $(\circ, \mp, \mathrm{Q}) \in \{(-, \epsilon, \bigcup), (+, \epsilon, \bigcap), (+, \neg, \bigcup), (-, \neg, \bigcap)\}$
and j occurs in Π

(5)

$$\frac{\langle \{\Gamma, \mathit{pref}^{\circ} \mp s\gamma, \Delta\}, \mathcal{A}\rangle}{\langle \{\mathit{pref}^{\circ} \mp s\gamma, \Gamma, \mathit{pref}^{\circ} \mp \overline{s\gamma}, \Delta\}, \mathcal{A}\rangle}$$

where $\overline{s\gamma}$ is the execution of the substitution s in the formula γ

Anna Zalewska
Logic, Informatics and Philosophy of Science Chair
Univ. of Białystok
Poland
e-mail: zalewska@hum.uwb.edu.pl