

**Axioms of a Set Theory**

Let  $\leq$  denote an ordering over sets, and  $\min(S)$  represent the least element of  $S$  (defined for all  $S \neq \emptyset$ ).

- 1] Firstly, all the axioms of first-order predicate calculus with equality.
- 2] The empty set ( $\emptyset$ ) exists.
- 3] Two sets are equal iff every member of one is also a member of the other.
- 4]  $\leq$  is a total order.
- 5] For every set  $S$ ,  $\emptyset \leq S$ .
- 6] For every set  $S$ ,  $S \leq \{S\}$
- 7] For any two sets  $S$  and  $R$ ,  $S \leq (S \cup R)$ .
- 8]  $S < R$  if  $\min(S) < \min(R) \vee (\min(S) = \min(R) \wedge (S \setminus \{\min(S)\} < (R \setminus \{\min(R)\}))$
- 9] If  $P$  is a valid set machine program, and  $S$  is a set (by these axioms), then the set  $O$  which is the output set of that set machine program given input  $S$ , exists.

**Set machines**

A set machine has a finite number of registers  $n$ , numbered  $r_0 \dots r_{n-1}$ ,  $n \geq 2$ . Each register can store a set, which exists according to the above axioms. Registers  $r_0$  and  $r_1$  are specially reserved:

- $r_0$  is defined to contain the output of the program. The output of the set machine program is defined to be the set containing as its elements the contents of  $r_0$  for every step of the program, starting from the first step in which  $r_0$  is written to. (This later proviso is necessary to be able to generate output sets not containing  $\emptyset$ .)
- $r_1$  is defined to contain the input of the program. The input can be any set which exists according to these axioms

Registers  $r_2 \dots r_{n-1}$  are for working storage. Initially, every working storage register contains  $\emptyset$ .

The set machine has a program which consists of a finite series of  $n$  instructions, numbered  $0 \dots n-1$ , and an instruction counter  $0 \leq c < n$ , which indicates the current instruction. After executing each instruction, the counter increments by one, unless that instruction was a jump; upon reaching  $n-1$ , it then wraps around back to  $0$ . Set machine programs never halt. (However, a set machine program with a finite output set is equivalent to a halting program.)

The set machine has a special register  $G$ , which may contain either a set, or the special value  $\infty$  (which is not a set), but which is defined such that for every set  $S$ ,  $S < \infty$ . Initially the value of  $G$  is  $\infty$ .

The following operations are defined:

- 0] UNION  $r_1 r_2 r_3 \equiv r_1 \leftarrow r_2 \cup r_3$
- 1] DIFF  $r_1 r_2 r_3 \equiv r_1 \leftarrow r_2 \setminus r_3$
- 2] SINGL  $r_1 r_2 \equiv r_1 \leftarrow \{r_2\}$
- 3] EQU  $r_1 r_2 i \equiv \text{if } r_1 = r_2 \text{ then } c \leftarrow i$
- 4] MIN  $r_1 r_2 \equiv \text{if } r_2 \neq \emptyset \text{ then } r_1 \leftarrow \min(r_2)$
- 5] INDUCT  $r_1 r_2 r_3 r_4 \equiv \text{if } r_3 \text{ represents a valid set program and } r_2 < G \text{ then } r_1 \leftarrow \text{the output set of } r_3 \text{ executed with input } r_4, \text{ with the contents of register } G \text{ set to } r_2 \text{ during its execution}$

Note that the above definition of INDUCT requires the mapping of set machine programs to sets, to provide the contents of the  $r_3$  register. One approach is to use a construction of the natural numbers as sets in conjunction with a Gödel numbering of programs.