I. Introduction

In his book, "A Discipline of Programming" Dijkstra introduces a simple non-deterministic language based on his idea of guarded commands. He then introduces the idea of weakest preconditions on states to give his language a semantics called the predicate transformer semantics. This is very well-suited to showing that systems can achieve certain goals, that is to questions of correctness, and the whole thing fits well within the paradigm of denotational semantics where we can regard predicates and conditions as a kind of continuation. (See [Stra], [Gor], [Mil], [Sto] for an account of continuations and denotational semantics, generally, and [de B], [de R], [Jen] for predicate transformer semantics in this style.)

On the other hand one can give the semantics of simple imperative languages using state transformation functions, and employing complete partial orders with a least element to handle divergence (nontermination). This kind of semantics can be considered as a direct abstraction from an operational semantics given, say, via an abstract machine. Within this framework, nondeterministic state transformations can be handled using powerdomains ([Plo2], [Smy]) which are a weak analogue of powersets for the complete partial orders under consideration.

The relation between these approaches was considered by de Roever who showed, in [de R], how, given a nondeterministic state-transformation, m, and a predicate, R, one could define the weakest-precondition, wp(m, R) of R, relative to m; this work employed Plotkin's powerdomain, based on the Egli-Milner order. He showed too that wp(m, R) was even continuous in m and R, and indicated that this enabled one to obtain the predicate transformer semantics (for a variant of Dijkstra's language including recursion) from the state transformation one. This was all fully worked out (among other things) by de Bakker in [de B]. Wand and Back have made other relevant and important contributions ([Wan], [Bac]) and these will be discussed later.

In the present paper we regard this work as showing a homomorphism from the state transformation view to the predicate transformer one. By refining the definitions a little we succeed in strengthening the homomorphism to an isomorphism (for a slight variant of Dijkstra's language). This involves, on the one hand, using the properties of predicate transformers, given by Dijkstra in chapter 3 of his book, to define the partial order of predicate transformers and, on the other hand, replacing the Egli-Milner order by the Smyth powerdomain.

However Dijkstra did not consider any particular language when introducing predicate transformers. Rather he used a general, somewhat informal, idea of
mechanism. He gave some discussion of the relation between this view and a
direct definition for the case of his language of guarded commands. We complete
this discussion by showing how programs in our variant of his language can be
regarded as mechanisms and so obtain another definition of their weakest precondition
semantics. By examining the relationship between programs considered as mechanisms
and their state transformation semantics we are able to demonstrate that both
definitions of the predicate transformer semantics come to the same thing, thereby,
in a sense, justifying Dijkstra's ideas.

In section 2 of this paper we present a formalisation of Dijkstra's mechanisms
using the well-known idea of a transition relation. In section 3 we introduce our
variant of Dijkstra's language of guarded commands and show how programs can act as
mechanisms by defining the appropriate transition relations; this is the operational
semantics of the language. In section 4 we introduce the partial order, PT, of
predicate transformers and use it to give the predicate transformer semantics of our
language. In section 5 we use Smyth's powerdomains to define the partial order, ST,
of state transformations and use that to give the other semantics for our language;
then we show that PT is isomorphic to ST (theorem 5.9) and this is even an
isomorphism of the two semantics (theorem 5.12).

In section 6 we show how the operational semantics relates to the state trans-
formation semantics (theorem 6.1) and use this result to demonstrate that the
direct definition of the predicate transformer semantics gives the same result as
the definition via the operational semantics (corollary 6.2).

Acknowledgements.
I thank Robin Milner and Willem de Roever for helpful comments and discussions,
and Dines Bjørner and Cliff Jones for encouraging me to write this paper.