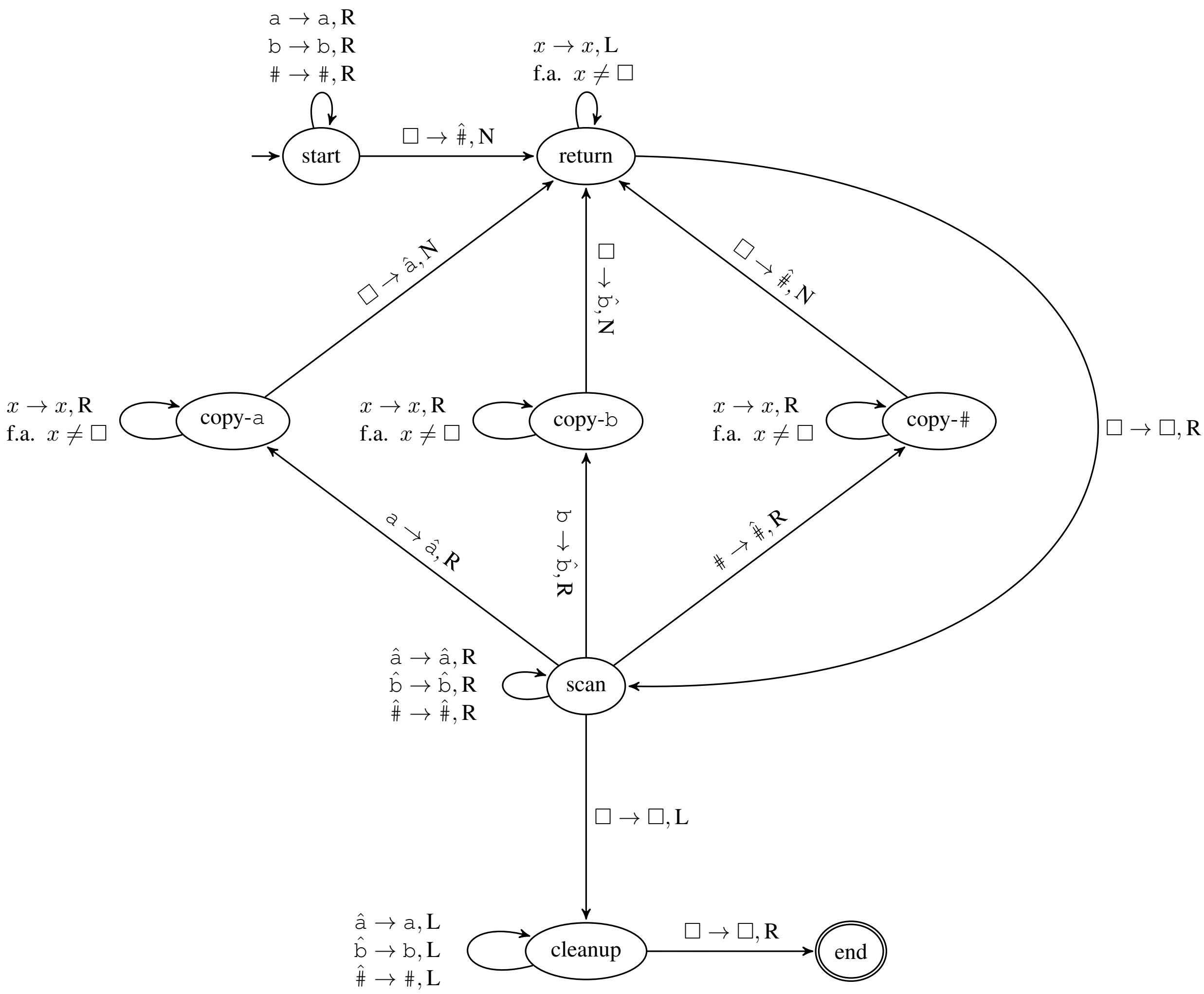
Theorie der Informatik

Florian Pommerening and Malte Helmert

FS 2016

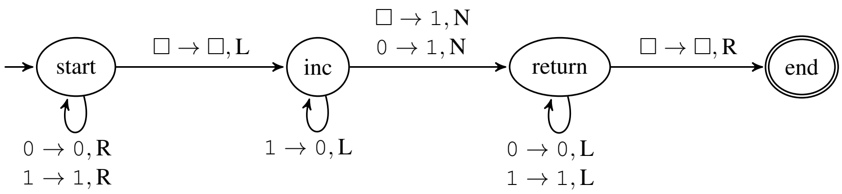
Notes for D1. Turing-Computability

Turing Machine for $w \mapsto w\#w$



In cases where no transitions are shown for a given non-end state and symbol (e.g., state "start" and symbol "â"), the design of the Turing machine guarantees that the corresponding transition is never taken during a computation, and hence it can be defined arbitrarily. These transitions are omitted for clarity of the picture. Formally, these transitions must exist to have a well-defined DTM, so we can for example define all such transitions to write the current symbol and remain in the current state (and hence loop forever).

Turing Machine for *succ* (First Attempt)



This Turing machine is almost correct. However, it only works correctly on well-formed inputs, i.e., inputs that correspond to a correct binary encoding of a single number. According to our definitions, the following kinds of ill-formed inputs are possible:

- Inputs that contain the symbol #. This symbol is formally part of our alphabet because it is needed for functions with multiple parameters. For unary functions like *succ*, all inputs including # are ill-formed.
- Inputs that only consist of symbols 0 and 1 but do not correctly encode a binary number. The only such inputs are the empty word ε and words beginning with the symbol 0 other than the word "0" (our encoding does not use leading zeros).

In order to match our definition, a complete Turing machine must ensure that on ill-formed inputs, it either does not terminate or terminates in an "invalid" configuration. The following Turing machine addresses these points.

Turing Machine for *succ* (Complete with Error Checking)