Dynamic Structures

Petri nets (2/2) - Advanced Reference Sheet

FMC

FMC diagrams for dynamic structures are based on transition-place Petri nets. They are used to express system behaviour over time depicting causal dependencies. So they clarify how a system is working and how communication takes place between different agents.

Here only the advanced notational elements are covered whereas the rest is located on the basic reference sheet (1/2).

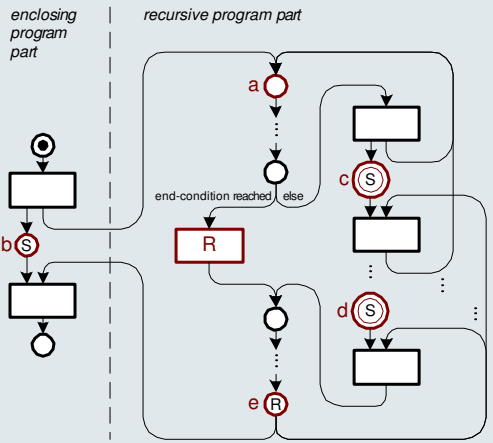
extended elements

	multi-token place	Places which can hold multiple tokens but not an infinite number are indicated as enlarged places with an annotation specifying the capacity (n>1). Places with an infinite capacity are indicated by a double circle.
	arc	The arc weight n determines how much tokens will be consumed or produced when the connected transition fires. An arc weight of 1 is assumed, if there is no one specified.

recursion elements

	stack place (cap.1, cap. infinite)	Is a place to store information about return positions using stack tokens. All stack places with the same name are strongly coupled with each other as the stack tokens, although placed on several stack places, are managed in a single stack. So all the stack places together constitute the return stack.
	return place	Is used like a normal place. But there is always a conflict to solve as a return place is an input place for at least two transitions that also have stack places as input places. When a return place gets a token and more than one associated stack places have a stack token the conflict is always solved in the same manner: the newest token on the stack must be consumed first. The newest token belongs to exactly one stack place and so the transition where this stack place is an input place will fire.

general recursion scheme



Each recursive diagram shows the following characteristics:

1) There is an entry point of the recursion (place a). Initially called by the enclosing program part it is called afterwards several times by the recursive program part itself.

2) Transition R represents the reaching of the end-condition, which is always present to finish the recursion by determining the function value of at least one argument without calling the recursive part again.

3) Stack places like b, c and d are always input places for transitions that additionally have a return place (e) as input. All the stack places together constitute the return stack which is used to store information about return positions.

4) A return place (e) is always input place for at least two transitions that also have stack places (b, c, d) as input places.

5) Be aware that the return stack's only task is to guide the recursion handling. In addition all the necessary data stack modifications like PUSH, POP and TOP have to be done to remember values such as intermediate results.