Master 1 international, Université de Nice Sophia-Antipolis

31/03/2014

Formal Models of Computation for Concurrency

Discussion sections

Marked Graph, Synchronous Data Flow and KRG

By Jean-Vivien Millo

Marked graph (Petri Net)

Marked graphs (MG) are a sub-class of Petri Net (PN) where places have exactly one incoming and one outgoing arc. Thanks to this restriction, MGs are conflict free.

Q1/ Draw a PN which is not a MG.

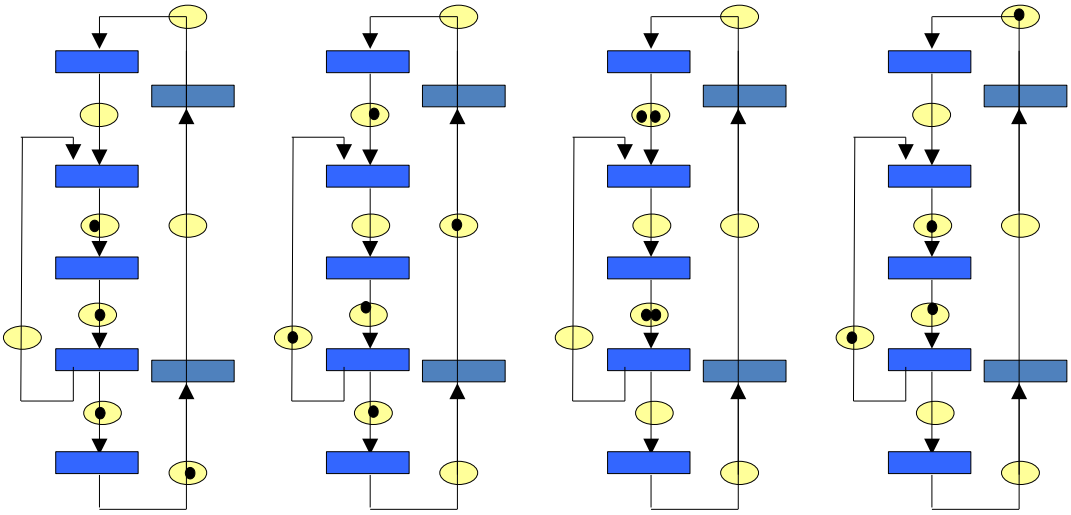
Q2/ Draw an MG which is i) not live, ii) live.

Q3/ Draw an MG which is not safe.

Q4/ Run the MG of Q3 with an ASAP scheduling. What do you see? How can you change the behavior so that the capacity of every place will be bounded?

Q5/ Draw an MG with a throughput 3/5.

Q6/ Consider the following MG and each of the four possible markings.



M1 M2 M3 M4

-Run the MG from M1 to M2.
-Run the MG from M1 to M3.
-Could you run the MG from M3 to M1 and M3 to M2?
-Could you run the MG from M3 to M4?
-Is there any invariant through the execution of these MGs?

Q7/ Run M1, M2, M3, and M4 with an ASAP scheduling until you reach the initial marking. While doing it, write the activation sequence (schedule) of each transition as a binary words (1=activity, 0=inactivity).

-What can you say about the obtained binary words?
-What can you say about the schedule of two successive transitions?
-What can you say about the schedules of M3? And M4?

Q8/ Consider that the application described by the marked graph given in Q6-M1 is executed on a quad core's architecture with shared memory.

-What is the best allocation of tasks (transitions) onto cores so that the throughput is maximized.

-How can you refine the marked graph in order to represents this allocation? And a specific schedule?

Synchronous Data Flow (SDF)

SDF is used to analyze data flow applications where the size of the computed data varies. For example: cryptography, error correcting code, audio and video processing…

SDF is a generalization of the MG model where the transitions consume and produce a fix amount of tokens. For example, each time a transition fires, it consumes three tokens on its left input, two on its right input and then produces four tokens on its output.

If the ratios are not correctly set up, the execution will lead to starvation or buffer overflow. Consequently, an SDF is considered as valid if and only if the ratios are balanced.

One can verify that an SDF graph is correct by building and solving the balance equations.

Q9/ Draw an SDF graph composed of four transitions with at least a cycle.

Q10/ Abstract this SDF graph in its topology matrix. Is it balanced? If not, modify it such that it is.

Q11/ How many times each transition should be fired such that each of them computes the same amount of data? How to compute these values?

Q12/ What is the necessary and sufficient condition to make your SDF live? Where one should put the tokens? How many?

-To think about this question, it is useful to transform the SDF graph in a homogeneous SDF graph (where each transition consumes and produces the same amount of token). Why is it useful?

K-periodic Routing Graph (KRG)

The KRG model is used to refine the SDF model by introduction deterministic choice using the select and merge node. The idea is to represent a larger set of applications specifically when data flows are split and/or merged. This is useful to represent bus based or NoC based communication, explicit parallelism, or even specific algorithmic details.

Similarly to SDF, KRG graphs are subject to starvation or buffer overflow in case of unbalanced ratio. In KRG this is also to the select and merge node. For example, if a select node sends twice more data on the left branch than the right one. The same ratio should be respected while merging the two flows.

Q15/ Draw a KRG composed of at least one select node, one merge node.

Q16/ Abstract the KRG in an SDF and check if it is well balanced

Q17/ Initialize the KRG such that it is live.

Q18/ Assume four tasks (A, B, C, D, and E), each task is allocated on its own core. The tasks communicate through a one-way ring bus. At every step, A sends a data to C and B sends a data to D. After ten steps, B and D send a data each to E while A and C stall. This pattern is repeated infinitely. We assume the places on the ring are delay free

-Represent the application using a KRG.
-Compute the switch condition of each of the merge and select node.
-Compute the schedule of each transition.