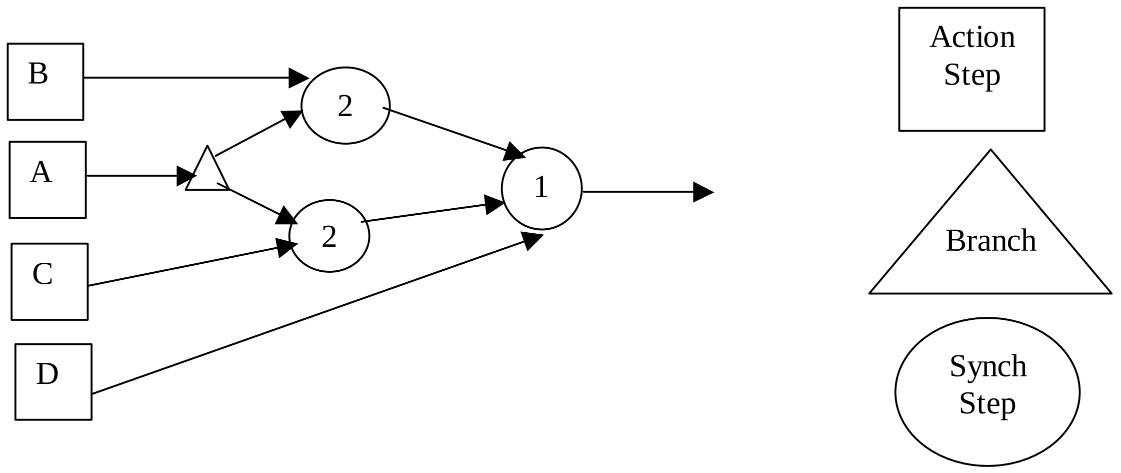
Synchronization and patient state steps

Patient_state_Step
    patient_state_description: Criterion
    next_step: Guideline_Step

Synchronization_Step
    minimum_incoming_arcs_to_continue: int
    next_step: Guideline_Step

Both patient state steps and synchronization steps can be used to synchronize.

Using Synchronization steps:
We link guideline steps with incoming arcs to the synchronization steps.
Using the attribute minimum_incoming_arcs_to_continue we cannot easily express logical combination of specific incoming guideline steps. For example, if we have Action Steps A,B,C and D and we want to synchronize them as (A and B) or (A and C) or D we will have to use a network of branch and synchronization steps:



We started thinking during the telephone meeting that instead of specifying only the minimum_incoming_arcs_to_continue we can have a logical expression that involves incoming Guideline_Steps with AND, OR, NOT operators. We even thought about having a continutation attribute of type Criterion (general criterion that can refer, for example to patient data).

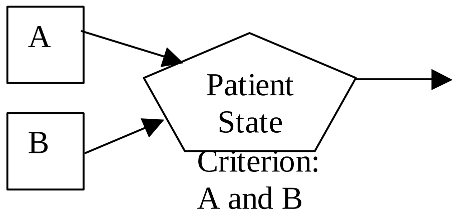
The problem is that by doing this we duplicate the functionality of the patient state step.

Synchronization steps are used to represent flow of control. They represent ordering constraints among guideline steps that have to hold. When we have an expression such as (A and B) or (A and C) or D, we mean that in order to proceed to the next step, we wait for the termination of both A and B, or both A and C, or just D. Using synchronization steps we have to put a connector between all previous guideline steps that need to be synchronized, and the synchronization step.

Using patient state steps for synchronization purposes:
Patient state steps can be used for labeling and as entry points into the guideline. When using them as labels, we say that when we arrive at a particular patient state then it means that we satisfy the criterion of the patient state step.

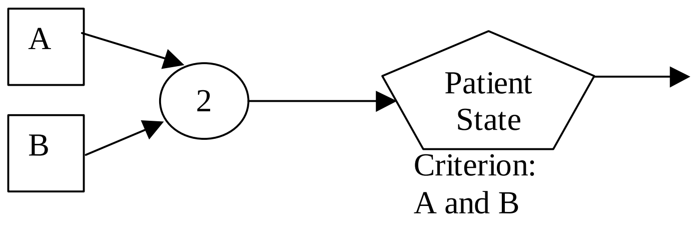
Patient state steps can also serve as entry points. When a patient state step represents an entry point it does not have to have incoming arcs. We never really defined the **semantics** of having incoming to a patient state that represents a new encounter. I think that when you arrive to such a patient sate then you remain in that state until the new encounter starts.

We never explicitly defined the semantics of a patient state step fully. For example



In this example we are at the patient state A and B when we are ready to leave it, that is, when both incoming arches fired.

Another alternative would be to say that as we enter a patient state then we are at that state. Then we would need to use a synchronization step as well:



Also, we don't have to link A and B at all to the patient state and can just leave it hanging there without incoming nodes. But this defeats the purpose of showing flow of control.

My personal opinion is that we should have both patient states and synchronization steps. The synchronization step will be used for the purpose of flow of control and the patient state for labeling and for entry points.

I propose changing the continuation attribute of the synchronization step to continuation: Logical_Expression_Of_Guideline_Steps

Where Logical_Expression_Of_Guideline_Steps is a structured string with the following production rule:

Logical_expression_of_guideline_steps: Guideline_Step | (Logical_expression_of_guideline_steps) | **not** Logical_expression_of_guideline_steps | Logical_expression_of_guideline_steps **and** Logical_expression_of_guideline_steps | Logical_expression_of_guideline_steps **or** Logical_expression_of_guideline_steps | **at least** Integer **of** Integer