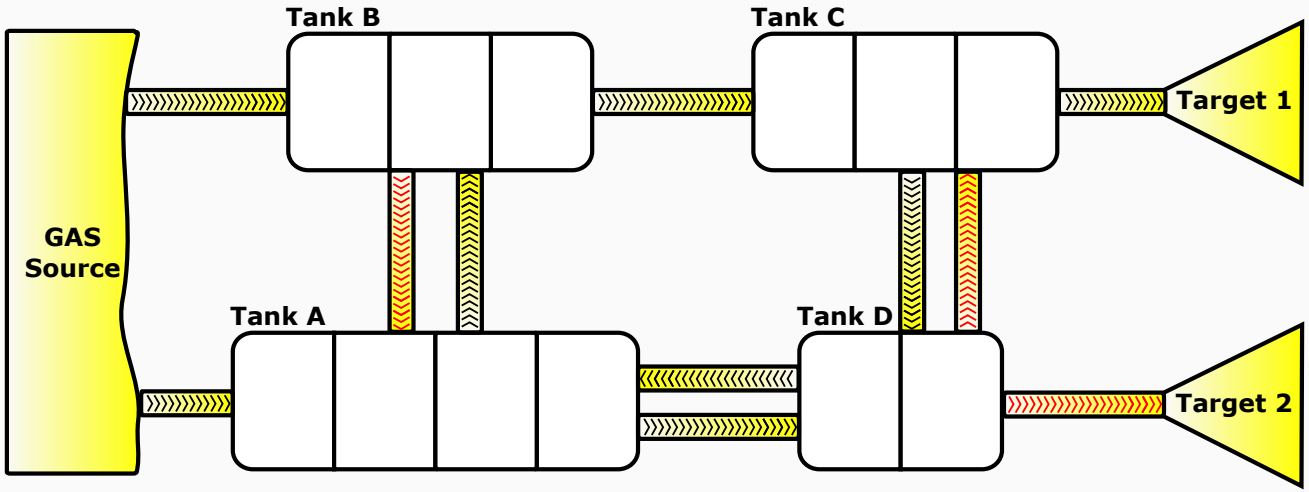
Systems Design Laboratory

Gas Tanks

Matteo Zavatteri

Department of Computer Science, University of Verona, ITALY

Plant

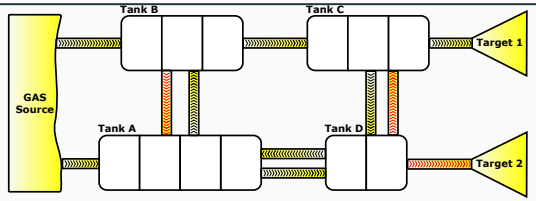


Main components:

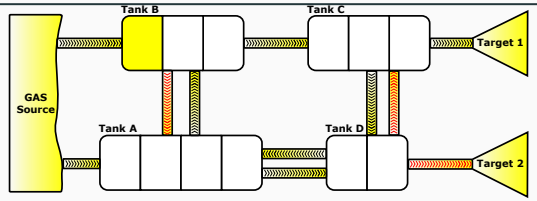
- An infinite source of gas
- 2 extraction points (leftmost pipes)
- 4 tanks of different capacity (given in units of gas, e.g., m^3)
- 2 target points of infinite capacity
- Several other pipes to move single units of gas from one tank to another according to the direction of the arrows.
- Red pipes always allow to move gas (if any) from their source to their target.

Usecase example

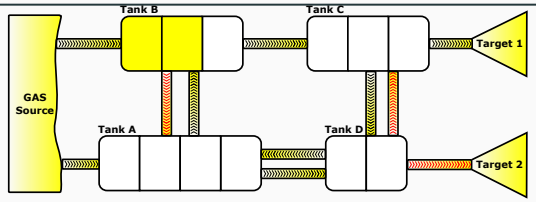
1) IDLE



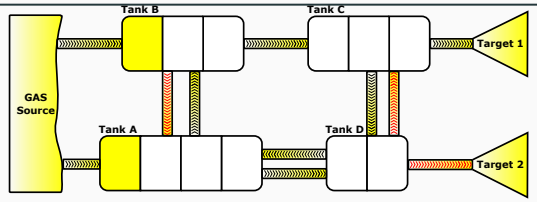
2) Source sends 1 gas unit to TankB



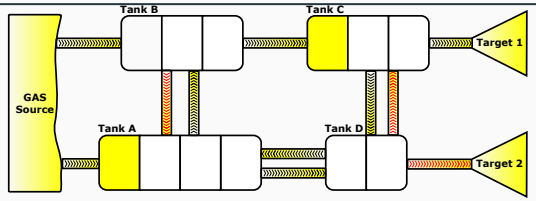
3) Source sends 1 gas unit to TankB



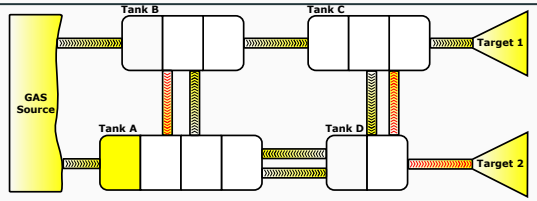
4) TankB sends 1 gas unit to TankA



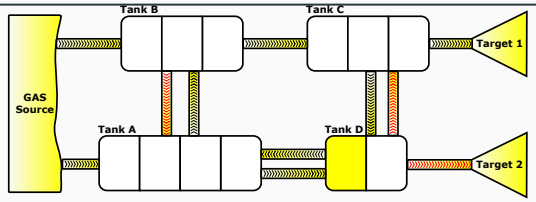
5) TankB sends 1 gas unit to TankC



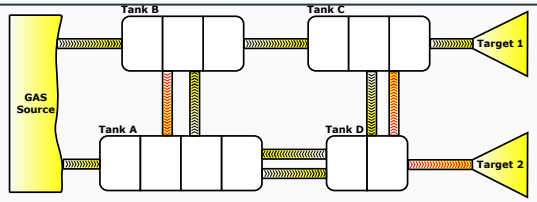
6) TankC sends 1 gas unit to Target 1



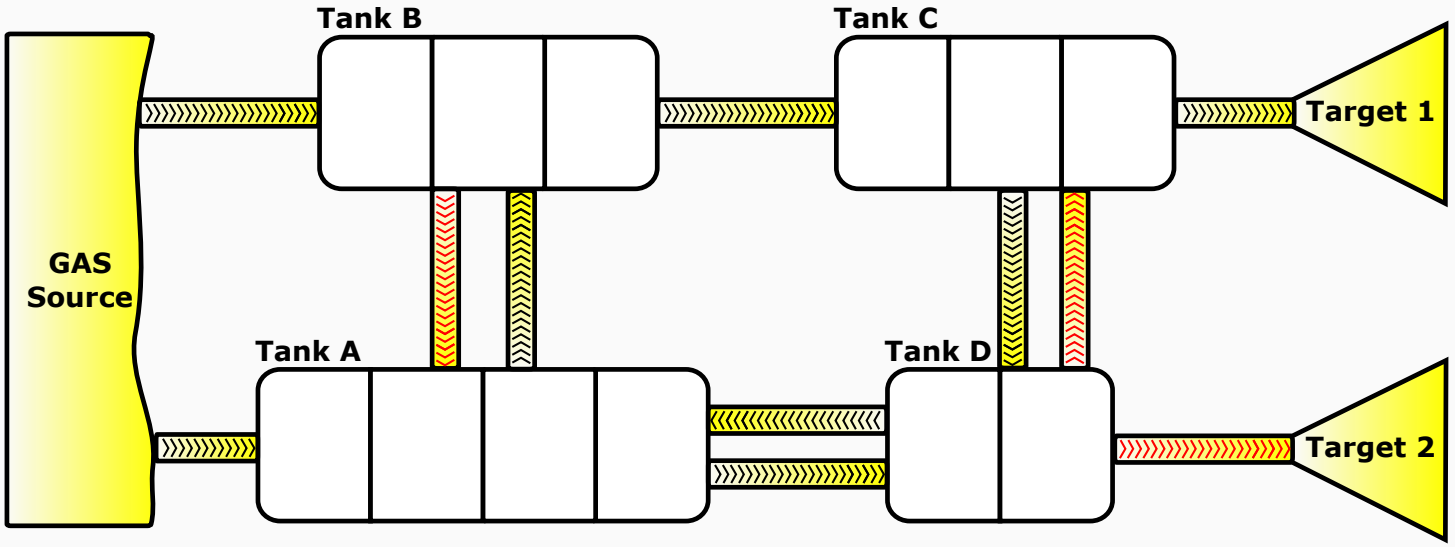
7) TankA sends 1 gas unit to TankD



8) TankD sends 1 gas unit to Target 2



The plant

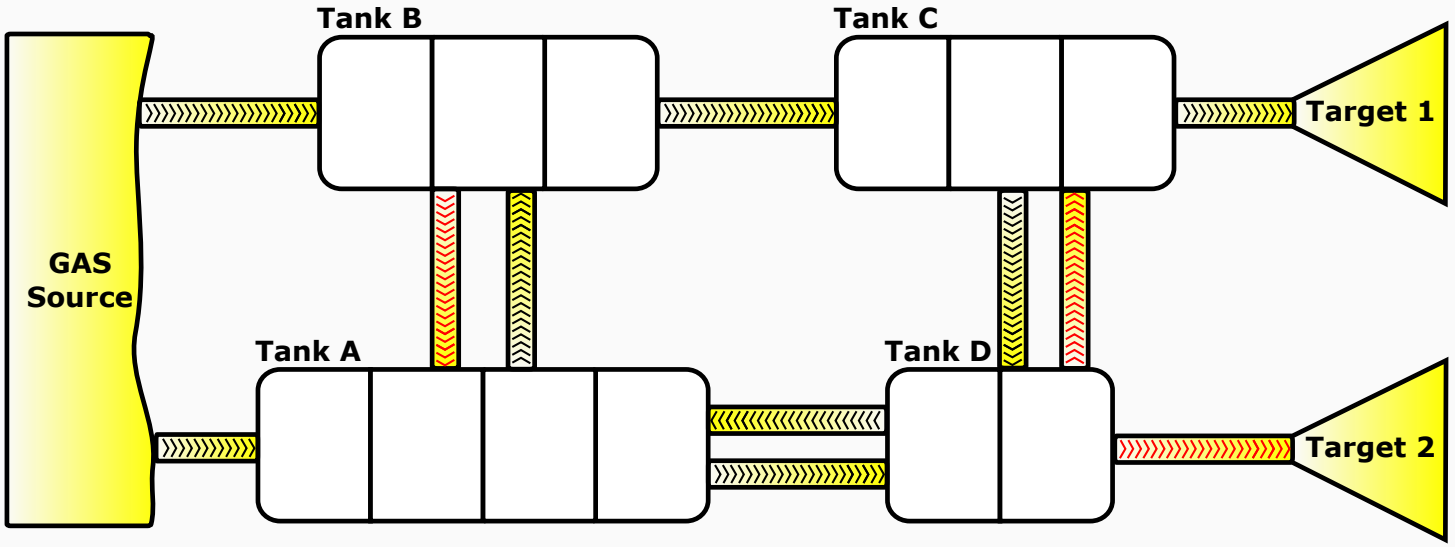


What should we model?

$\Downarrow$

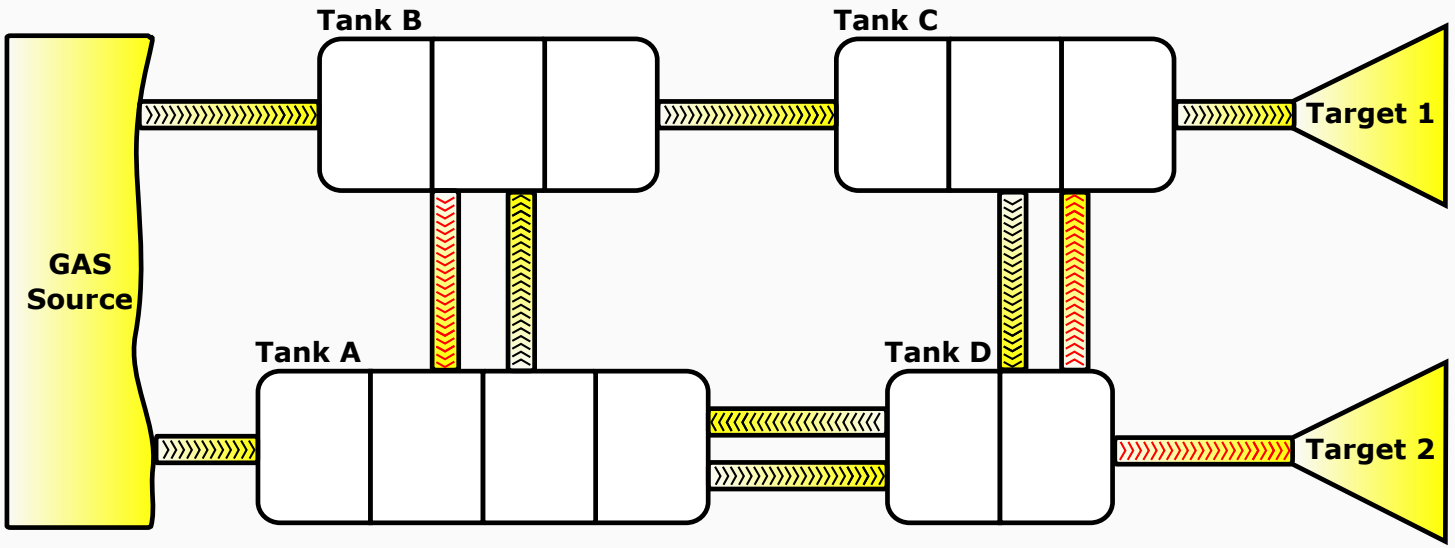
How many components?

Tank A



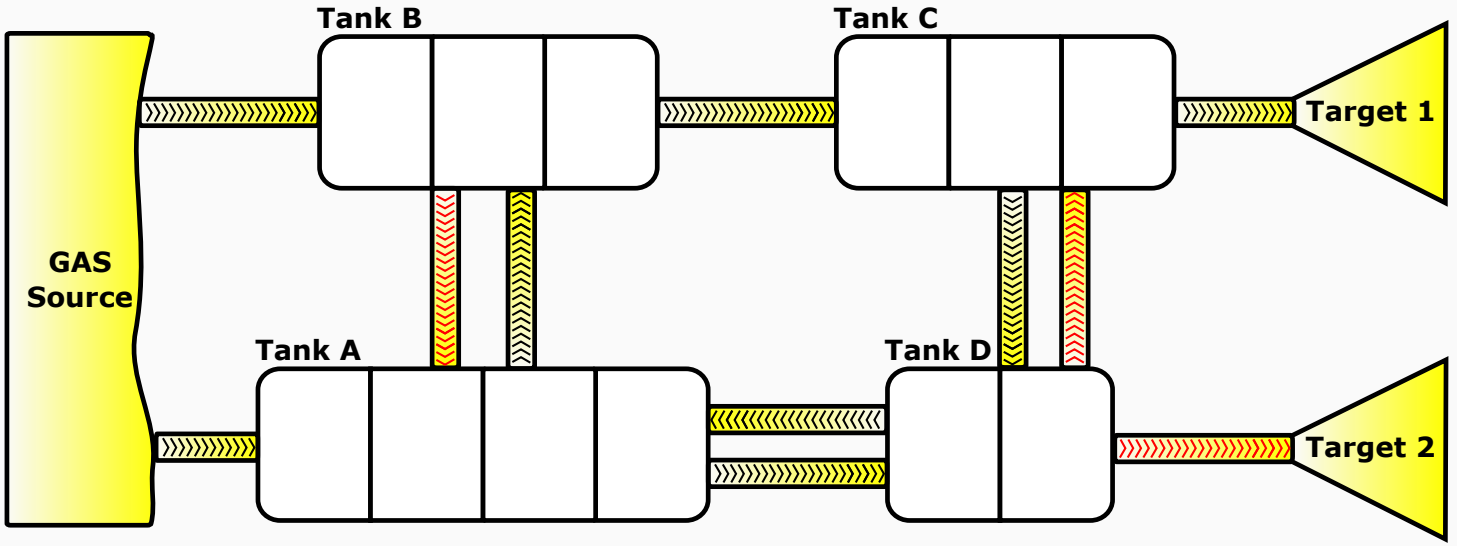
- States?
- Transitions?
- Event controllability?

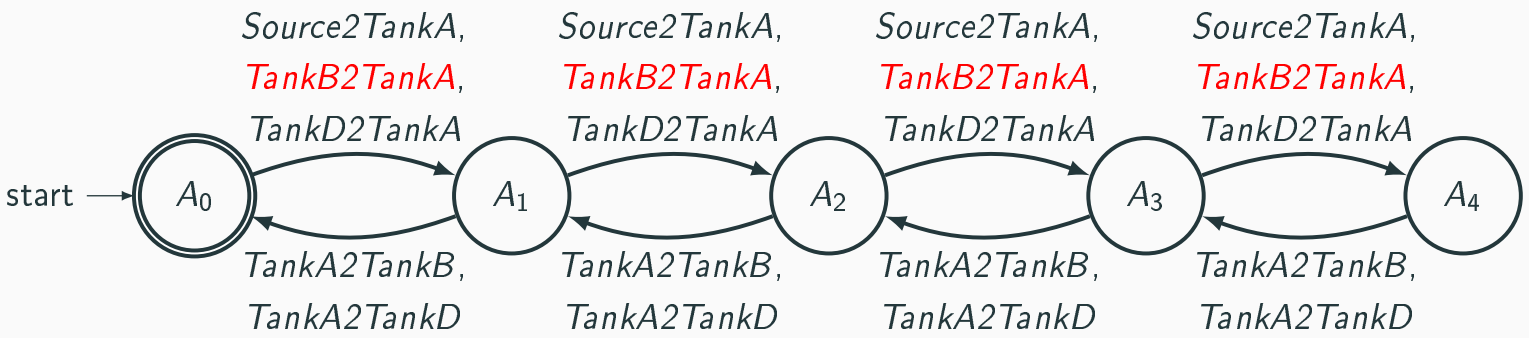
Tank A


Tank B
Tank C
Target 1
GAS
Source
Tank A
Tank D
Target 2


start
A_0
A_1
A_2
A_3
A_4

Tank A



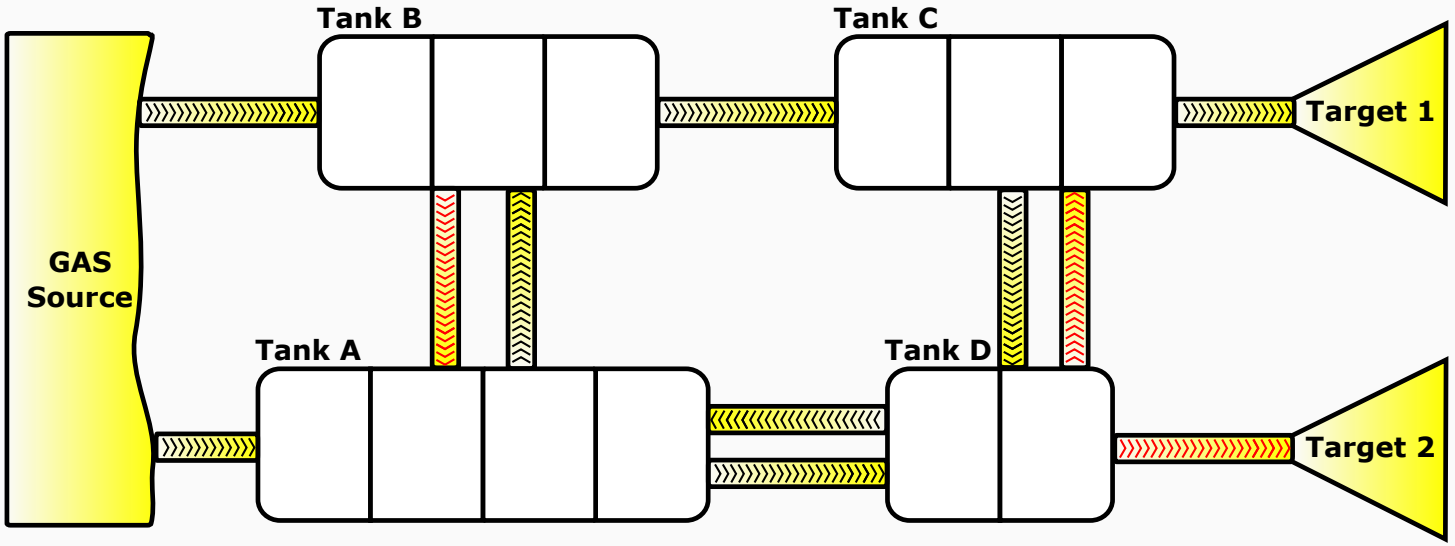


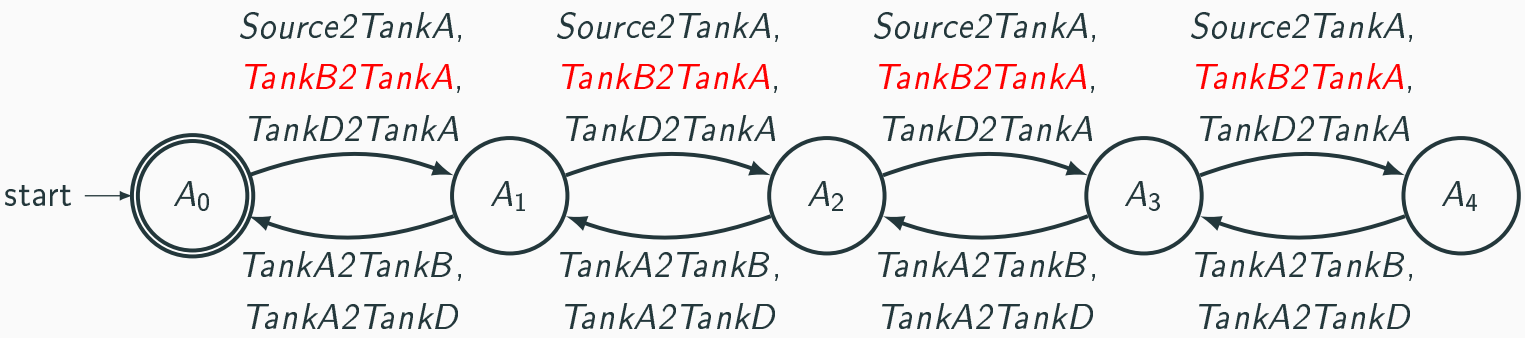
Event modeling intuition:

Source2TankA: **source supplies 1 gas unit to TankA**
TankB2TankA: **TankB supplies 1 gas unit, by removing it from itself, to TankA**

...

Tank A

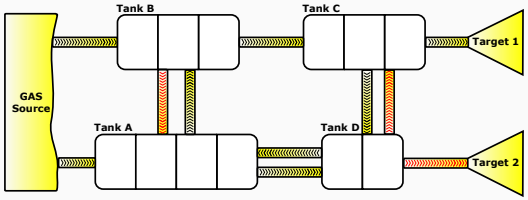




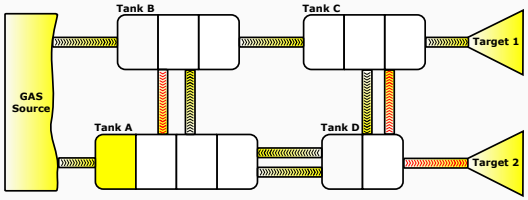
Realistic? Can't you spot a problem?

Realistically, without control might be like...

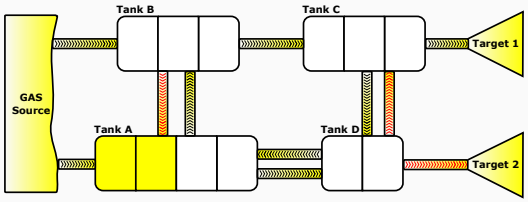
1) IDLE



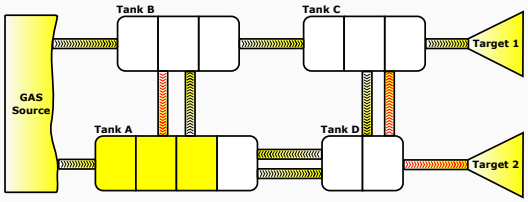
2) Source sends 1 gas unit to TankA



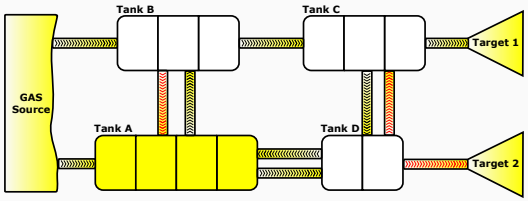
3) Source sends 1 gas unit to TankA



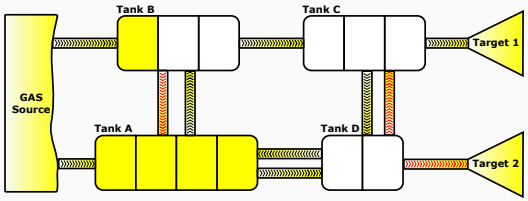
4) Source sends 1 gas unit to TankA



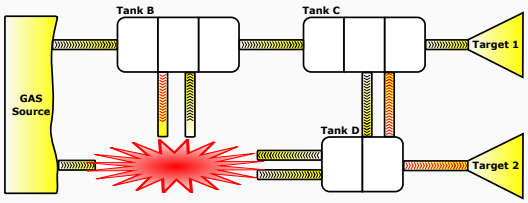
5) Source sends 1 gas unit to TankA



6) Source sends 1 gas unit to TankB

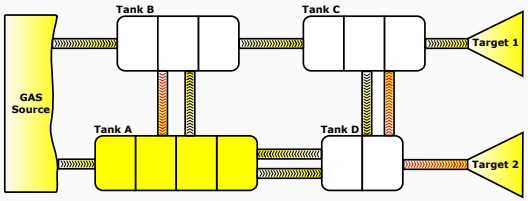
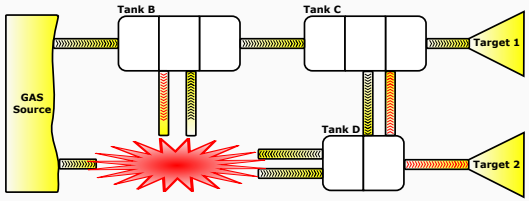
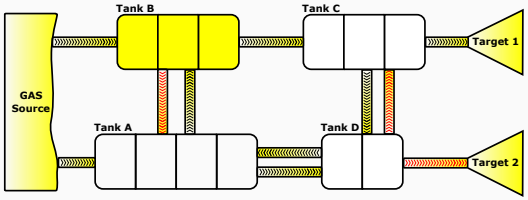
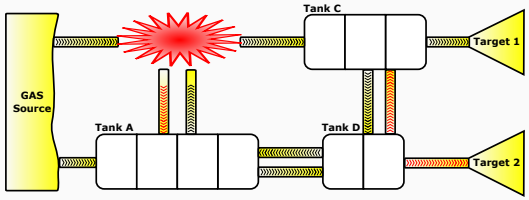
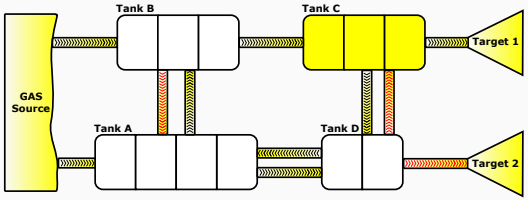
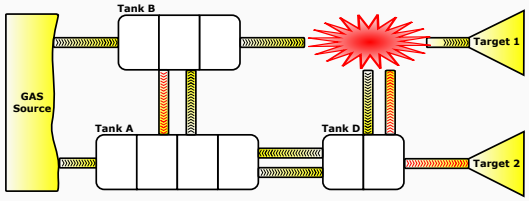
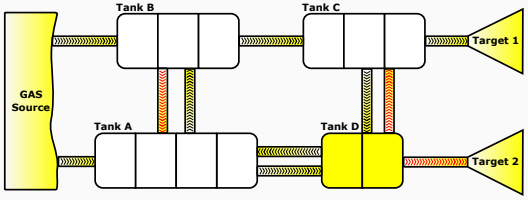
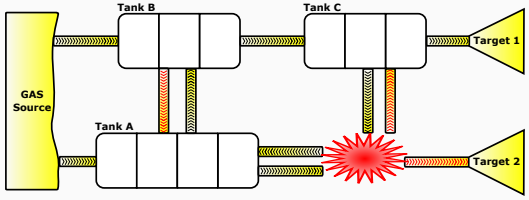


7) TankB sends 1 gas unit to TankA

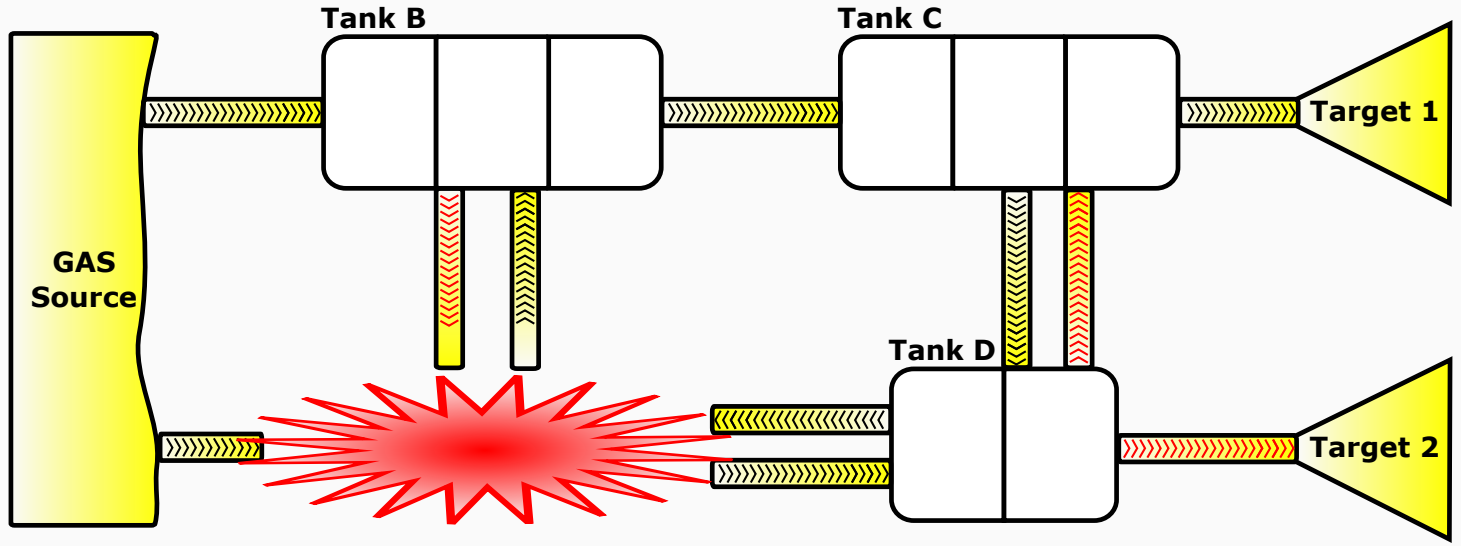


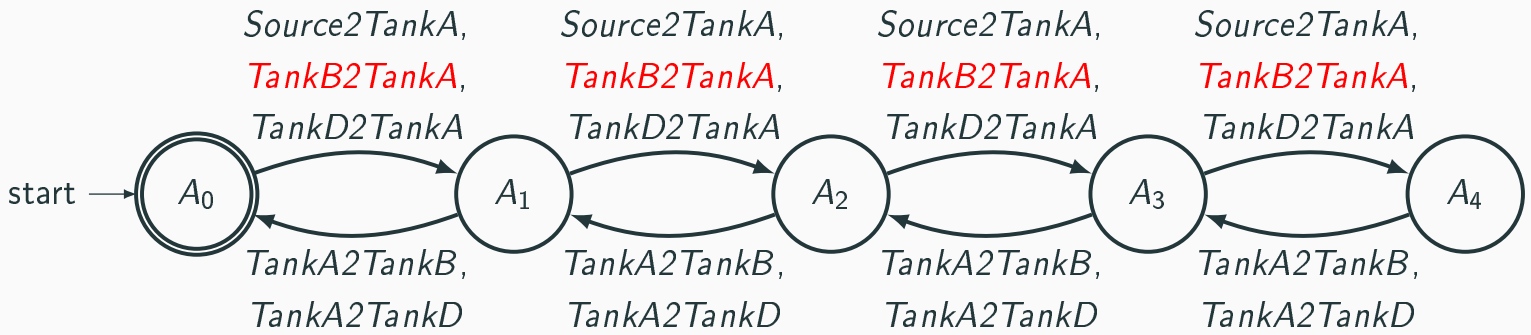
TankA blew up because it exceeded capacity. Note that *TankB2TankA* is uncontrollable.

Problem: If a tank exceeds its capacity, it blows up

Critical state	Condition(s) leading to explosion
TankA 	**If Source/TankB/TankD supplies TankA...** 
TankB 	**If Source/TankA supplies TankB...** 
TankC 	**If TankB/TankD supplies TankC...** 
TankD 	**If TankA/TankC supplies TankD...** 

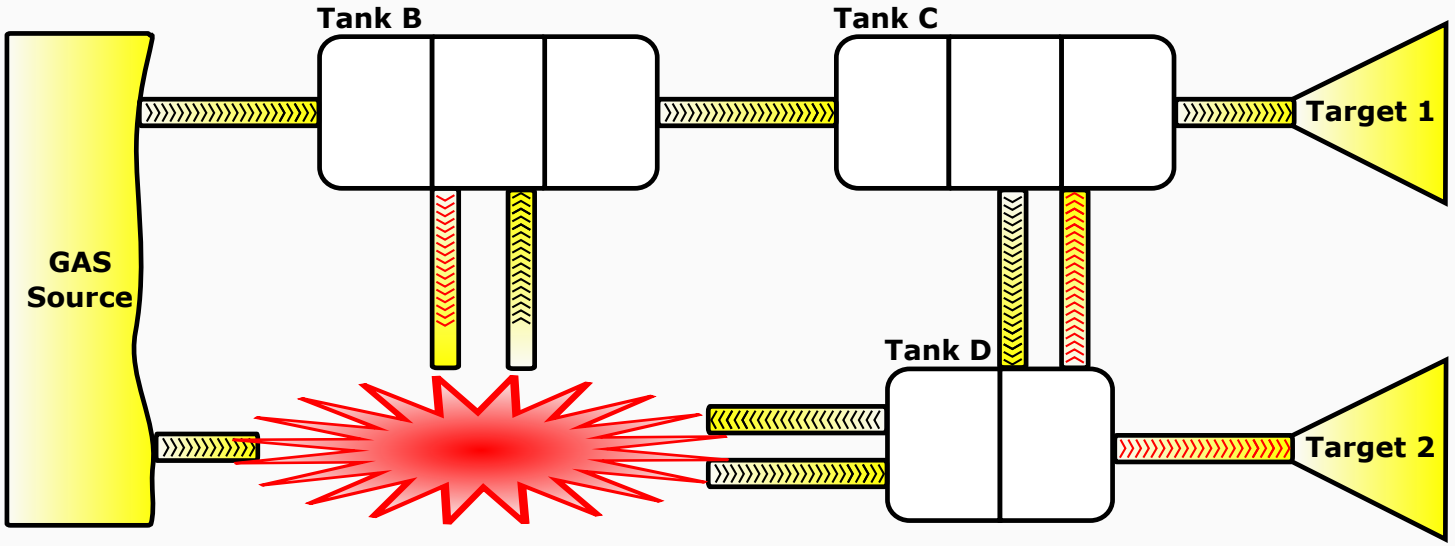
Tank A - Modeling the blow up situation

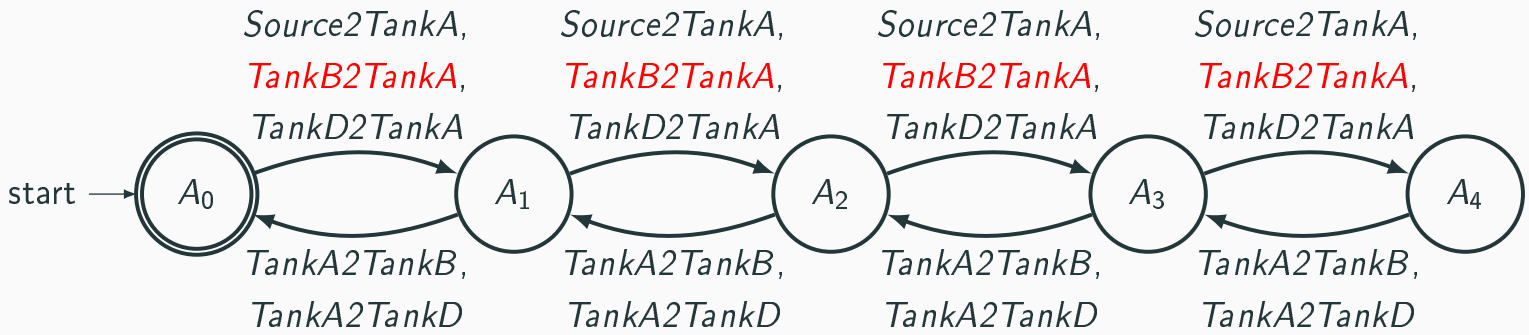




How can we model the fact that Tank A blows up?

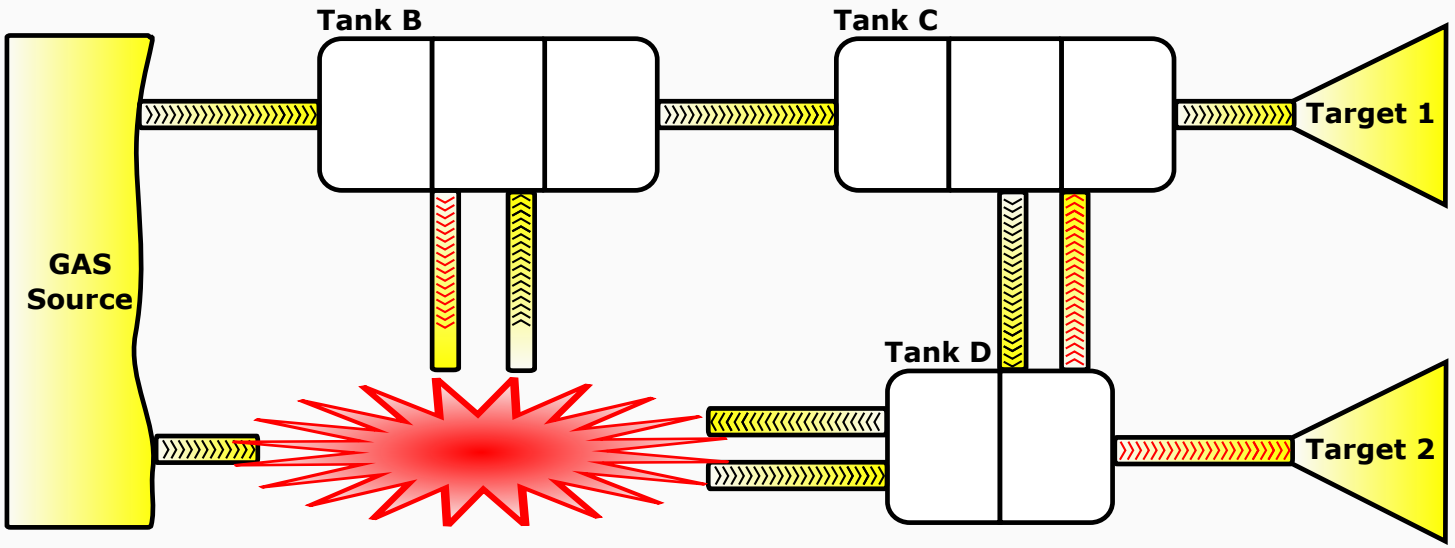
Tank A

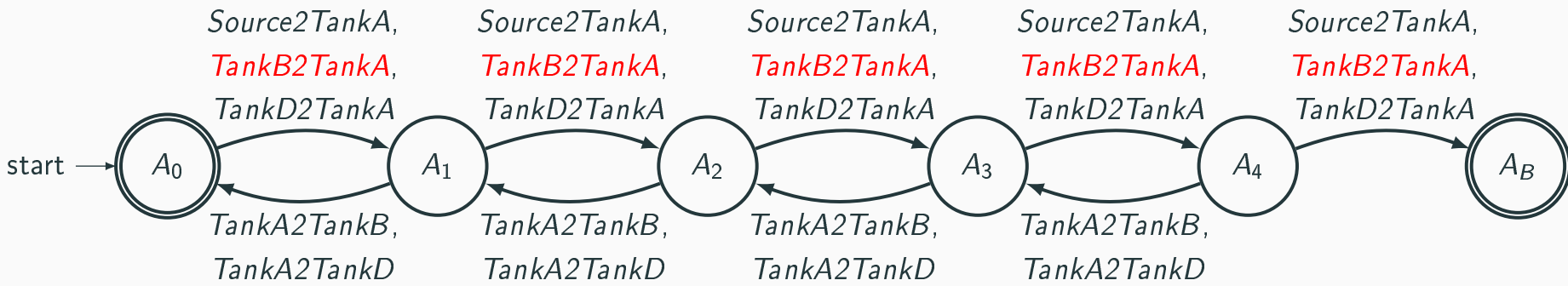




A new marked state to model "TankA has blown up"

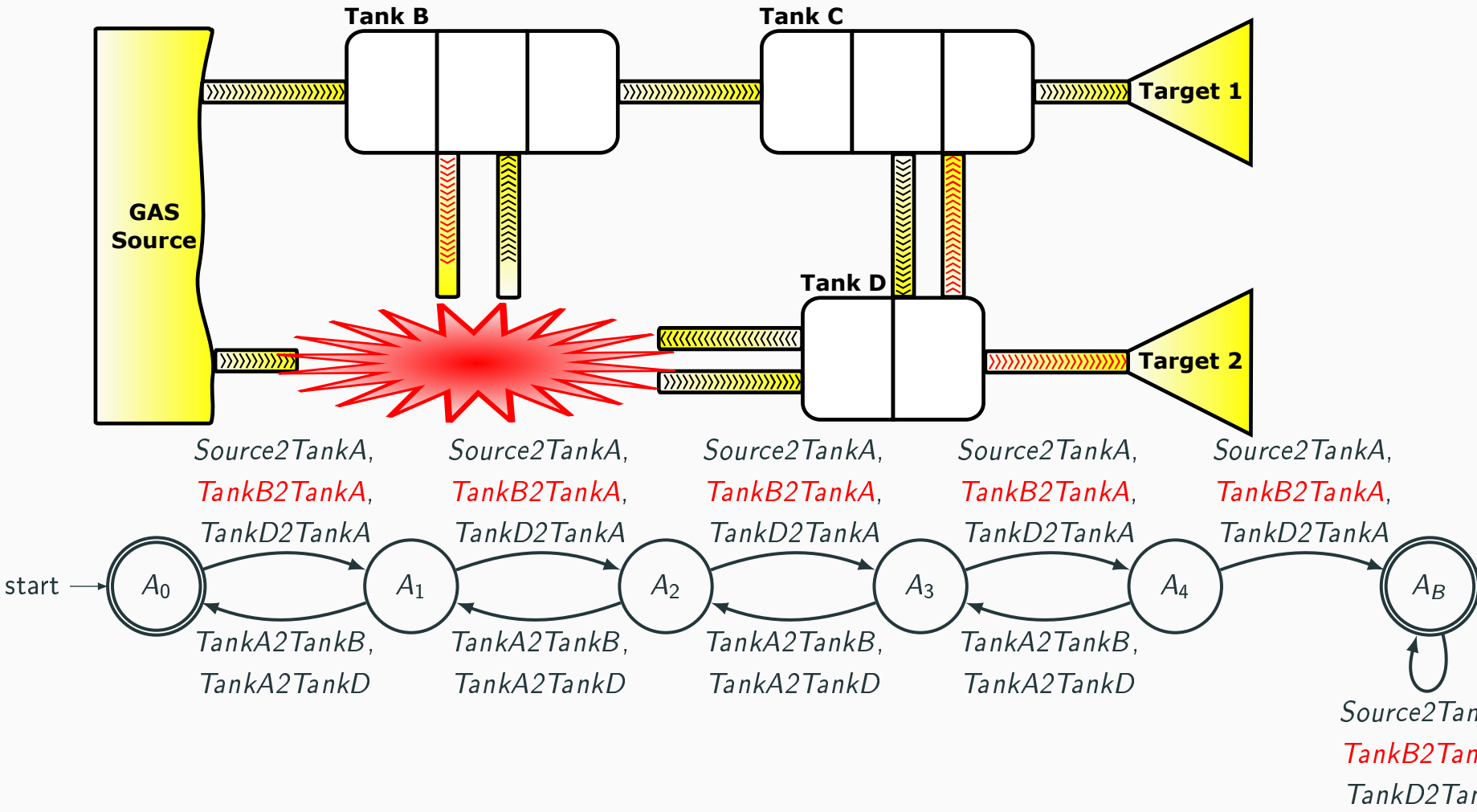
Tank A





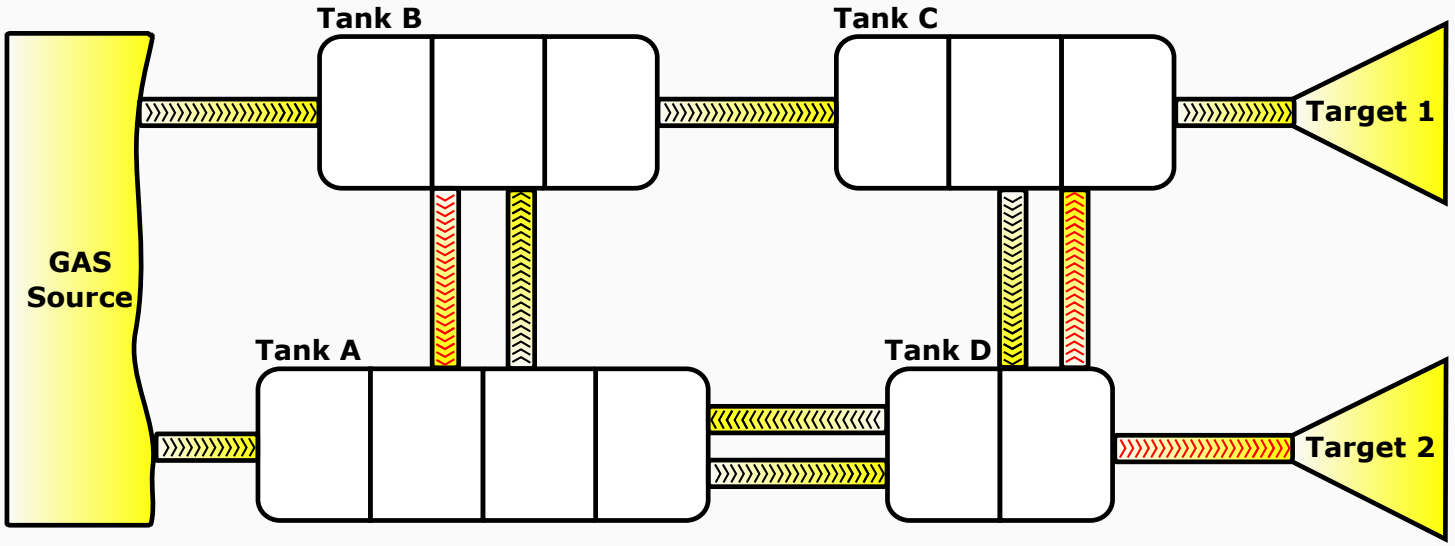
A_B is reachable if TankA exceeds its capacity.

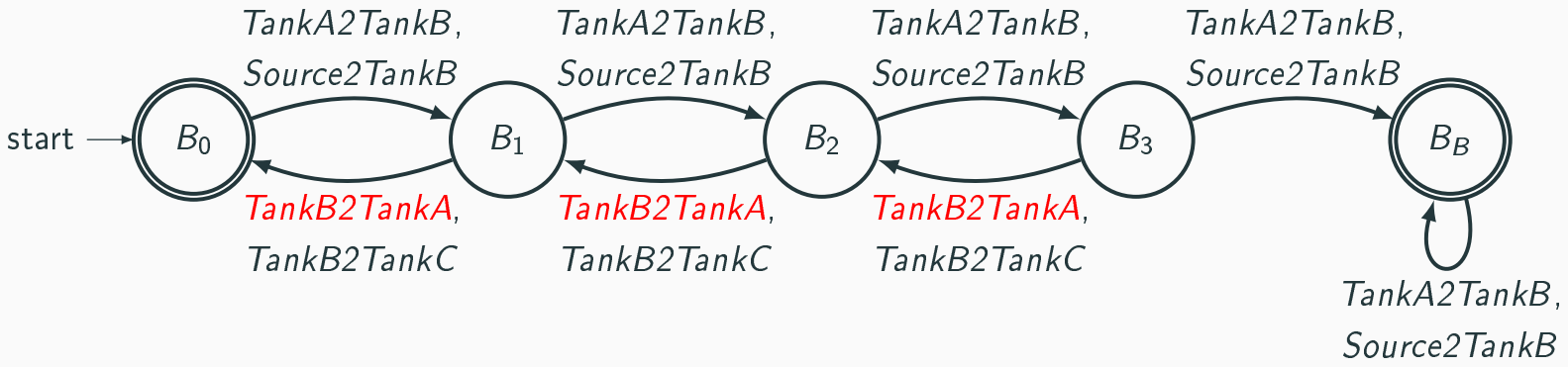
Tank A



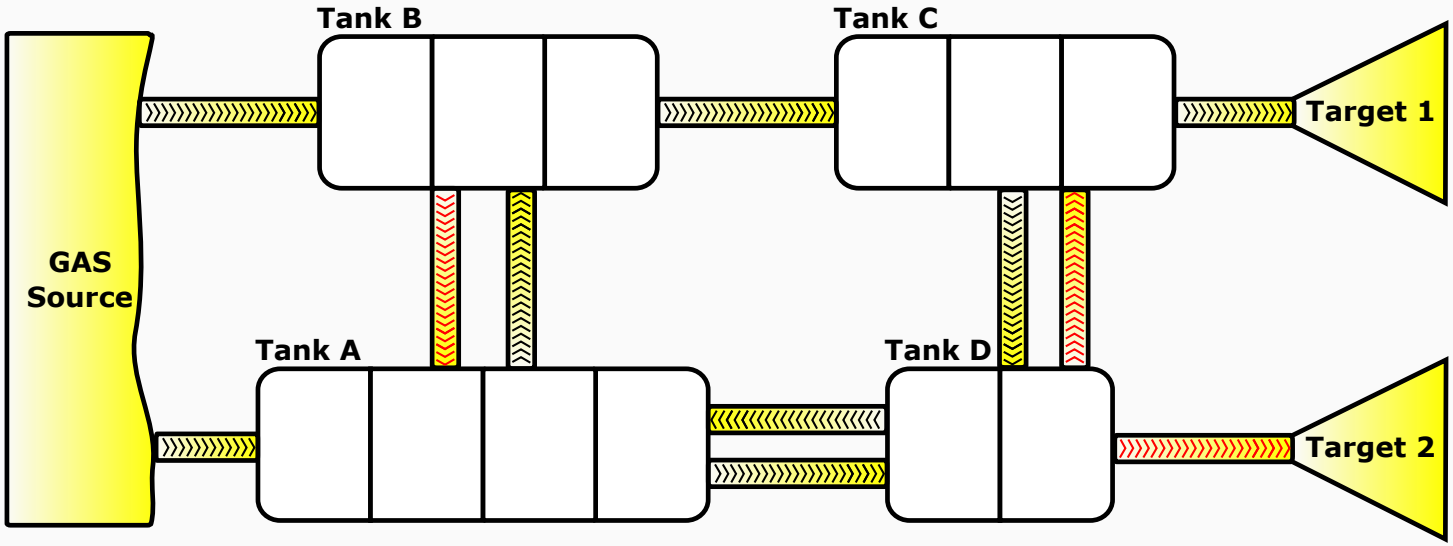
Once TankA blows up, the pipes sending gas to it can still send gas. If this happens, the shipped units of gas are clearly lost.

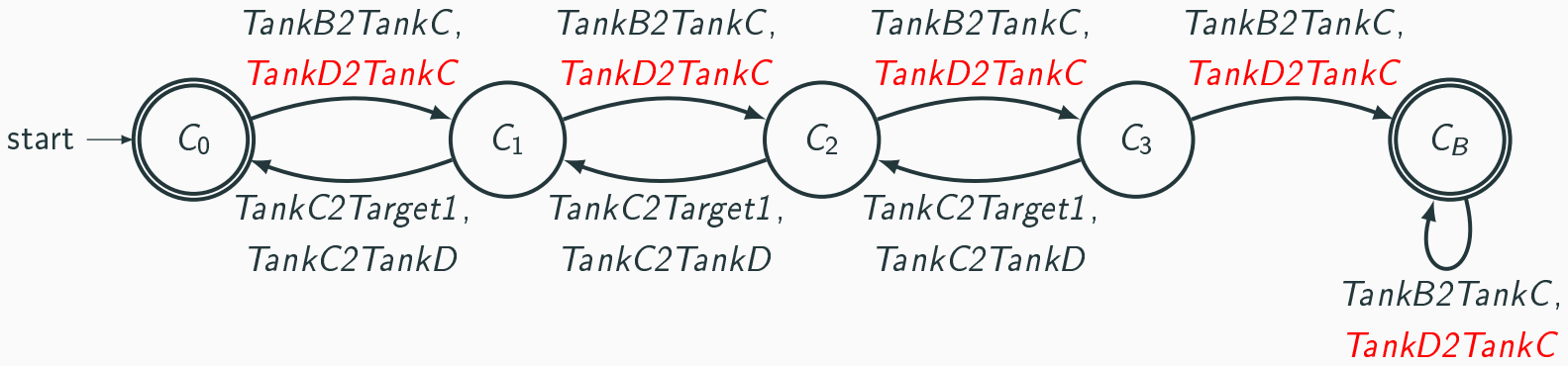
Tank B


Tank B
Tank C
Target 1
GAS
Source
Tank A
Tank D
Target 2

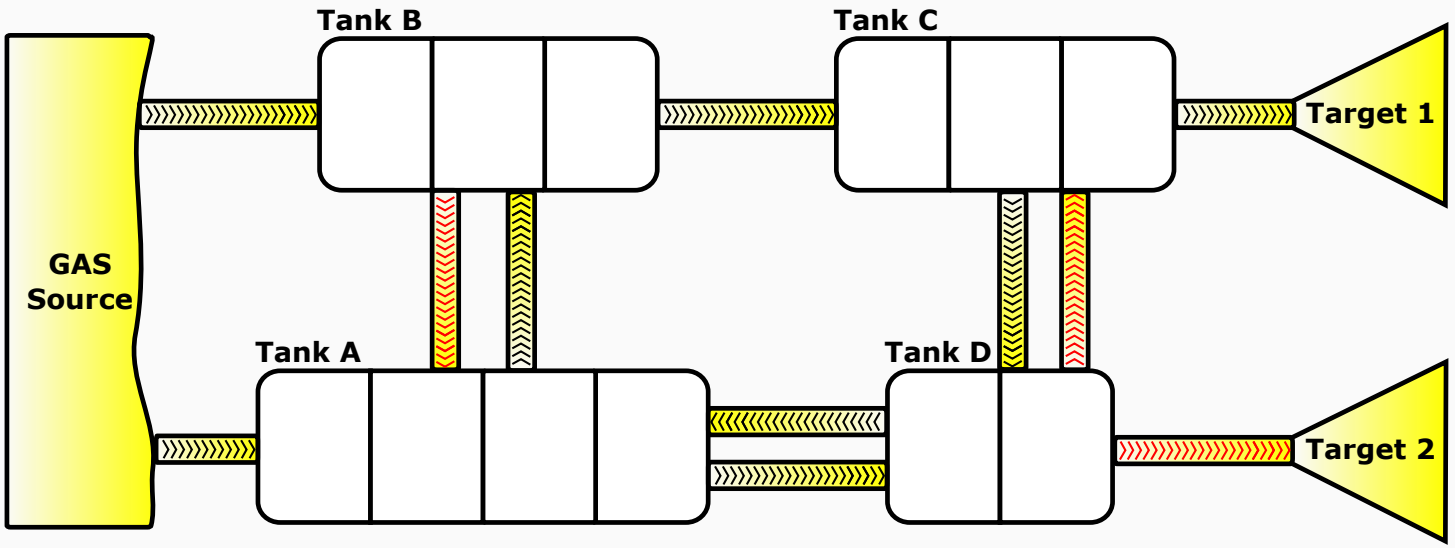

start
B_0
B_1
B_2
B_3
B_B
TankA2TankB,
Source2TankB
TankB2TankA,
TankB2TankC
TankA2TankB,
Source2TankB
TankB2TankA,
TankB2TankC
TankA2TankB,
Source2TankB
TankB2TankA,
TankB2TankC
TankA2TankB,
Source2TankB
TankA2TankB,
Source2TankB

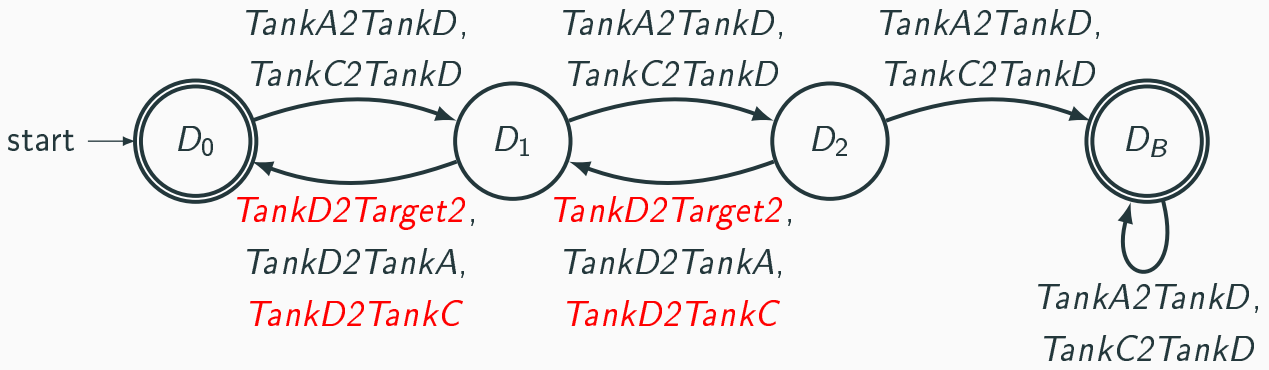
Tank C


Tank B
Tank C
Target 1
GAS
Source
Tank A
Tank D
Target 2

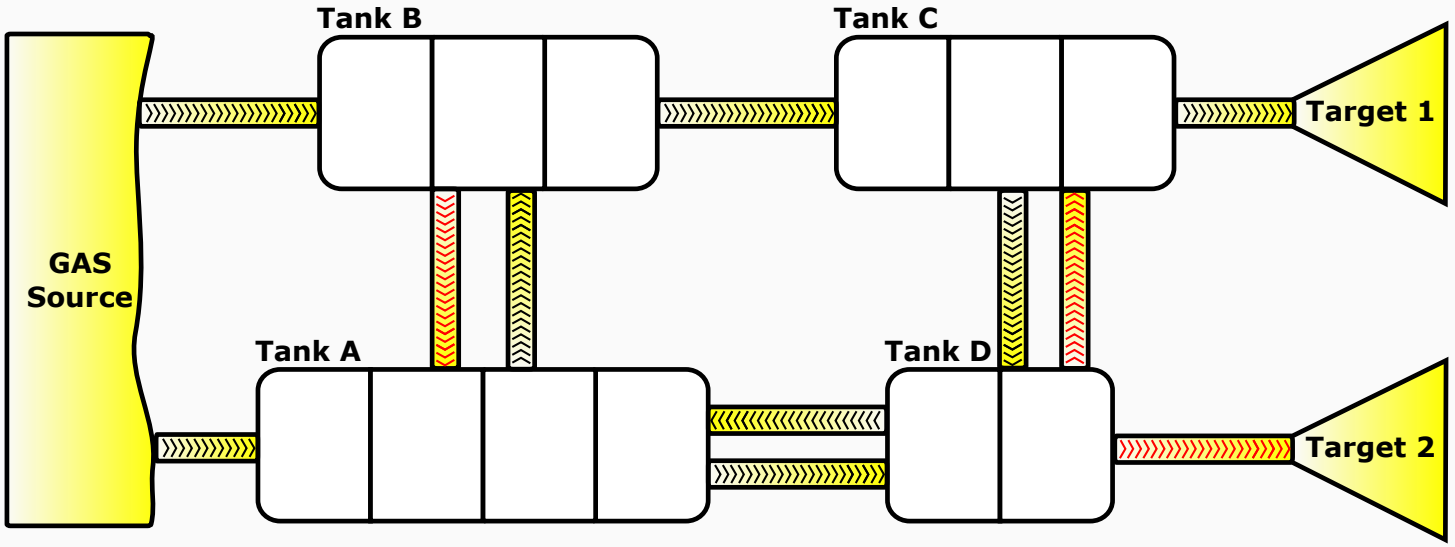

start
C_0
C_1
C_2
C_3
C_B
TankB2TankC,
TankD2TankC
TankC2Target1,
TankC2TankD
TankB2TankC,
TankD2TankC
TankC2Target1,
TankC2TankD
TankB2TankC,
TankD2TankC
TankC2Target1,
TankC2TankD
TankB2TankC,
TankD2TankC
TankB2TankC,
TankD2TankC

Tank D


Tank B
Tank C
Target 1
GAS
Source
Tank A
Tank D
Target 2

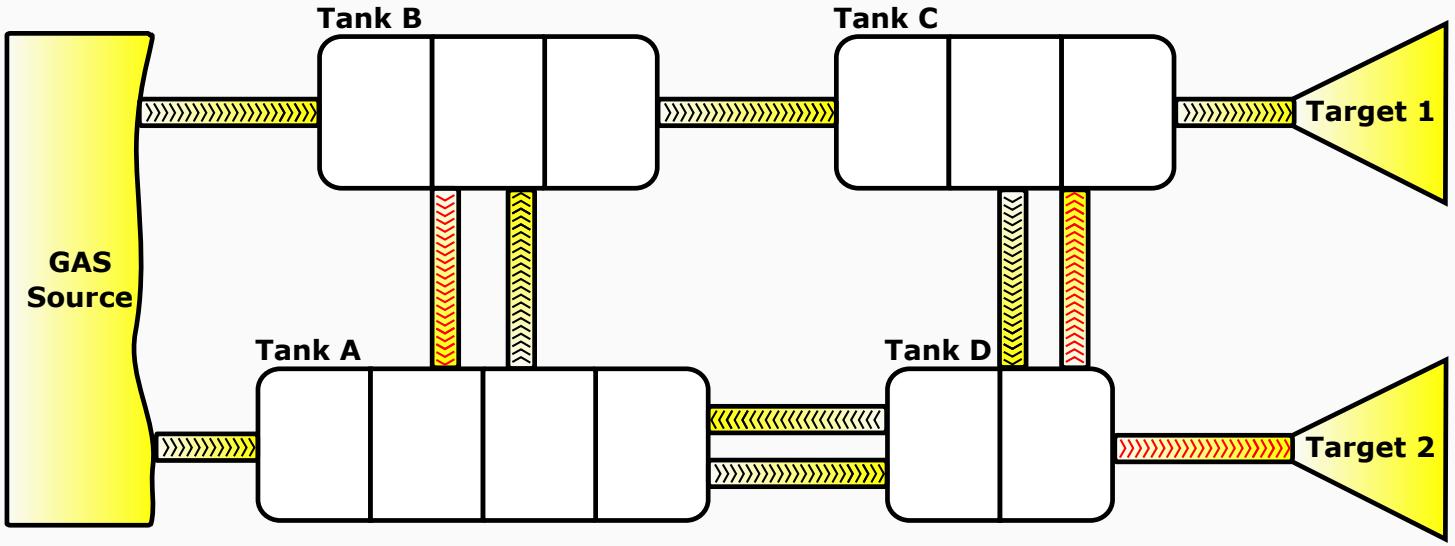

start
D_0
D_1
D_2
D_B
TankA2TankD,
TankC2TankD
TankA2TankD,
TankC2TankD
TankA2TankD,
TankC2TankD
TankD2Target2,
TankD2TankA,
TankD2TankC
TankD2Target2,
TankD2TankA,
TankD2TankC
TankA2TankD,
TankC2TankD

Source and Targets



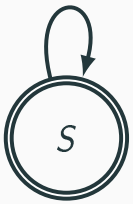
- States?
- Transitions?
- Event controllability?

Source and Targets



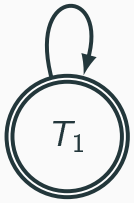
Source

Source2TankA, Source2TankB



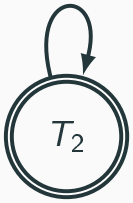
Target 1

TankC2Target1



Target 2

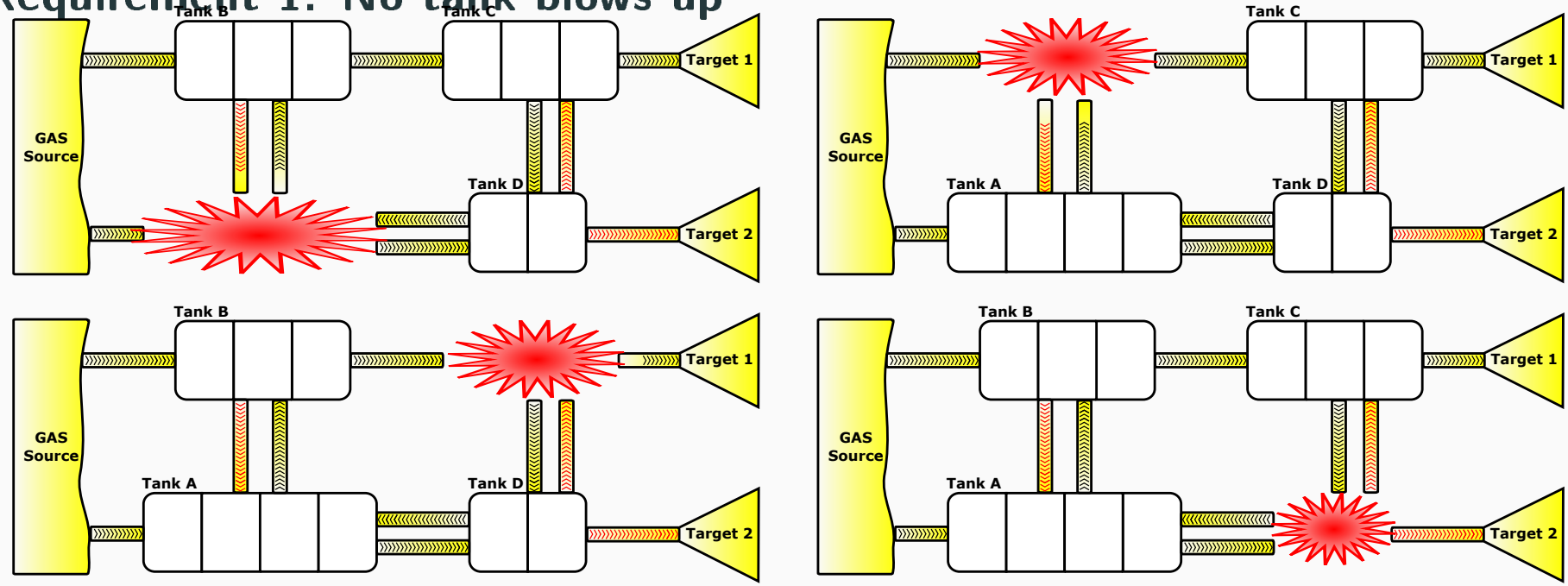
TankD2Target2



Not wrong, but quite useless.

Requirement 1

Requirement 1: No tank blows up



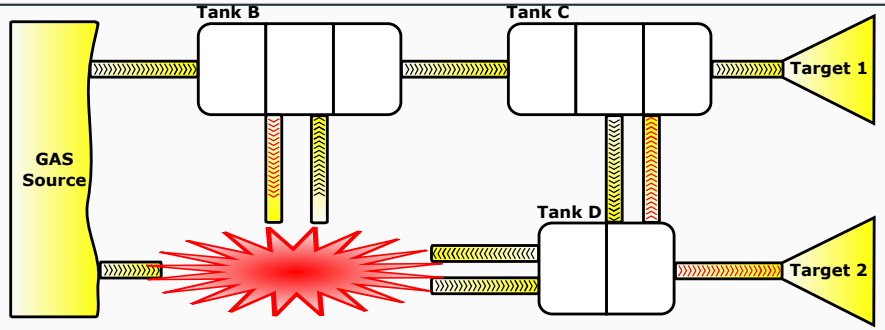
We may start from the parallel composition of all tanks ... (don't!)

Think: How many (separate) situations do we need to address?

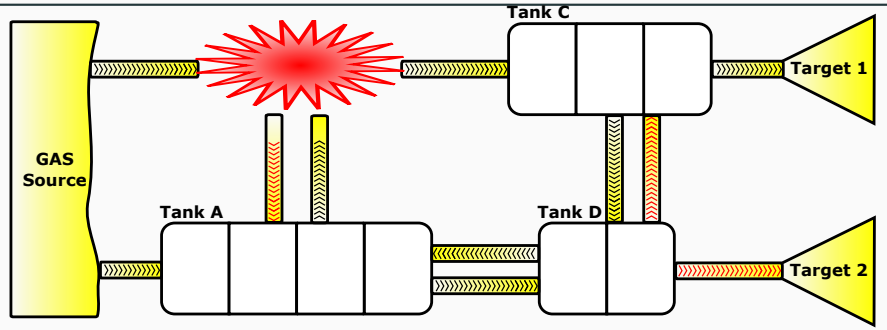
Requirement 1 - Decomposition

Requirement 1: No tank blows up

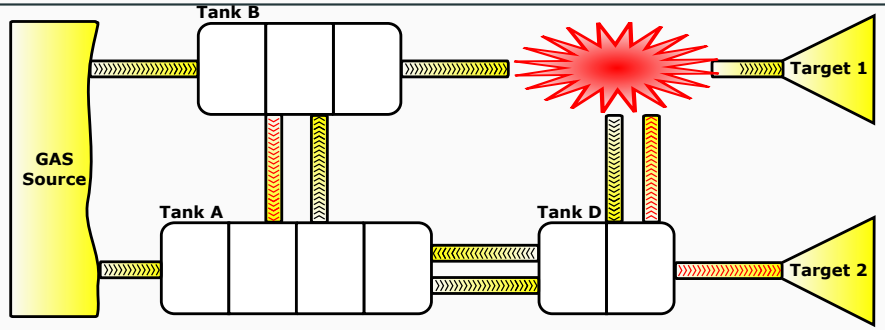
R_{1A}: TankA does not blow up



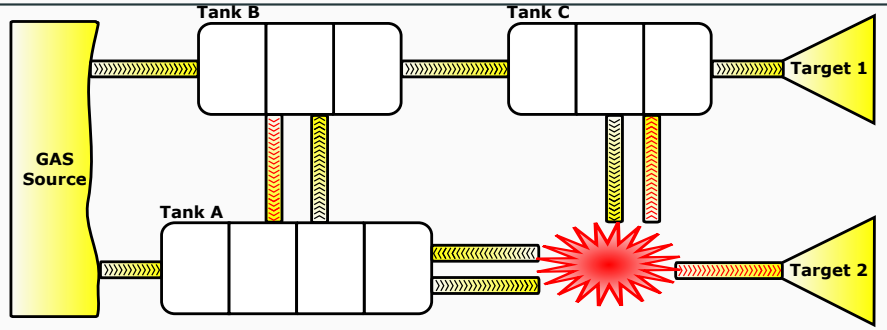
R_{1B}: TankB does not blow up



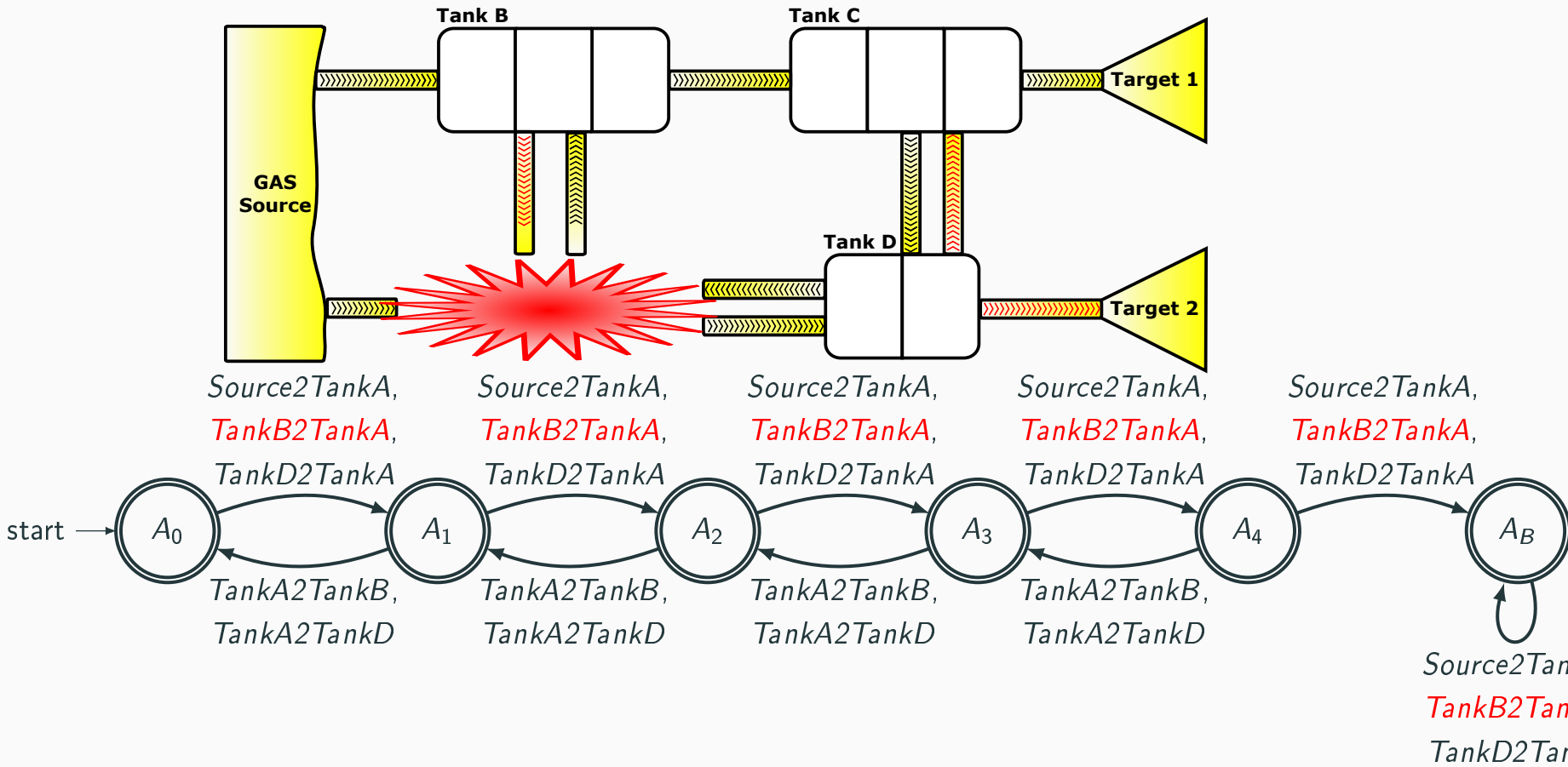
R_{1C}: TankC does not blow up



R_{1D}: TankD does not blow up

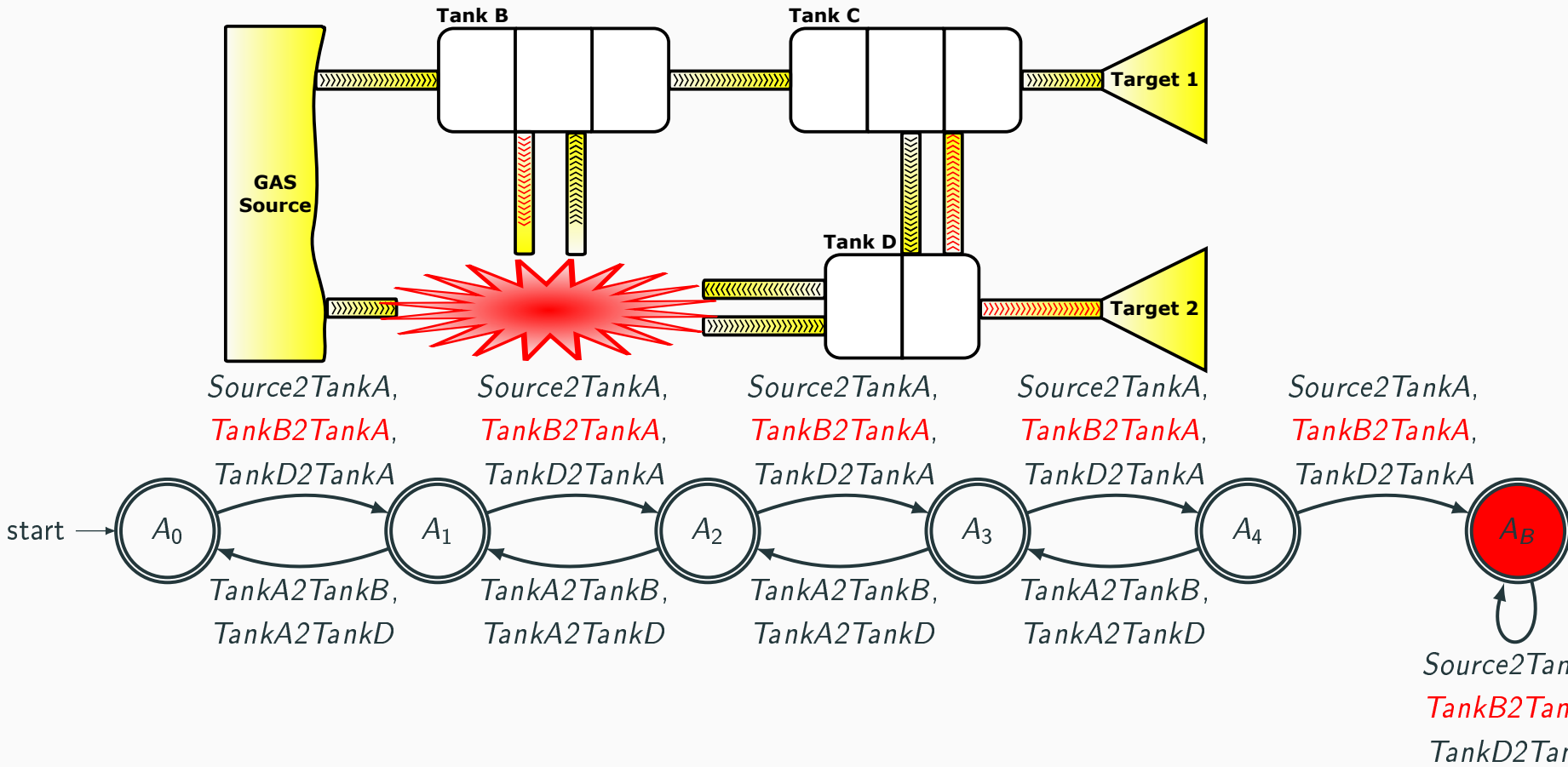


Requirement 1A: TankA does not blow up



Step 1: We need to keep track of the level of TankA

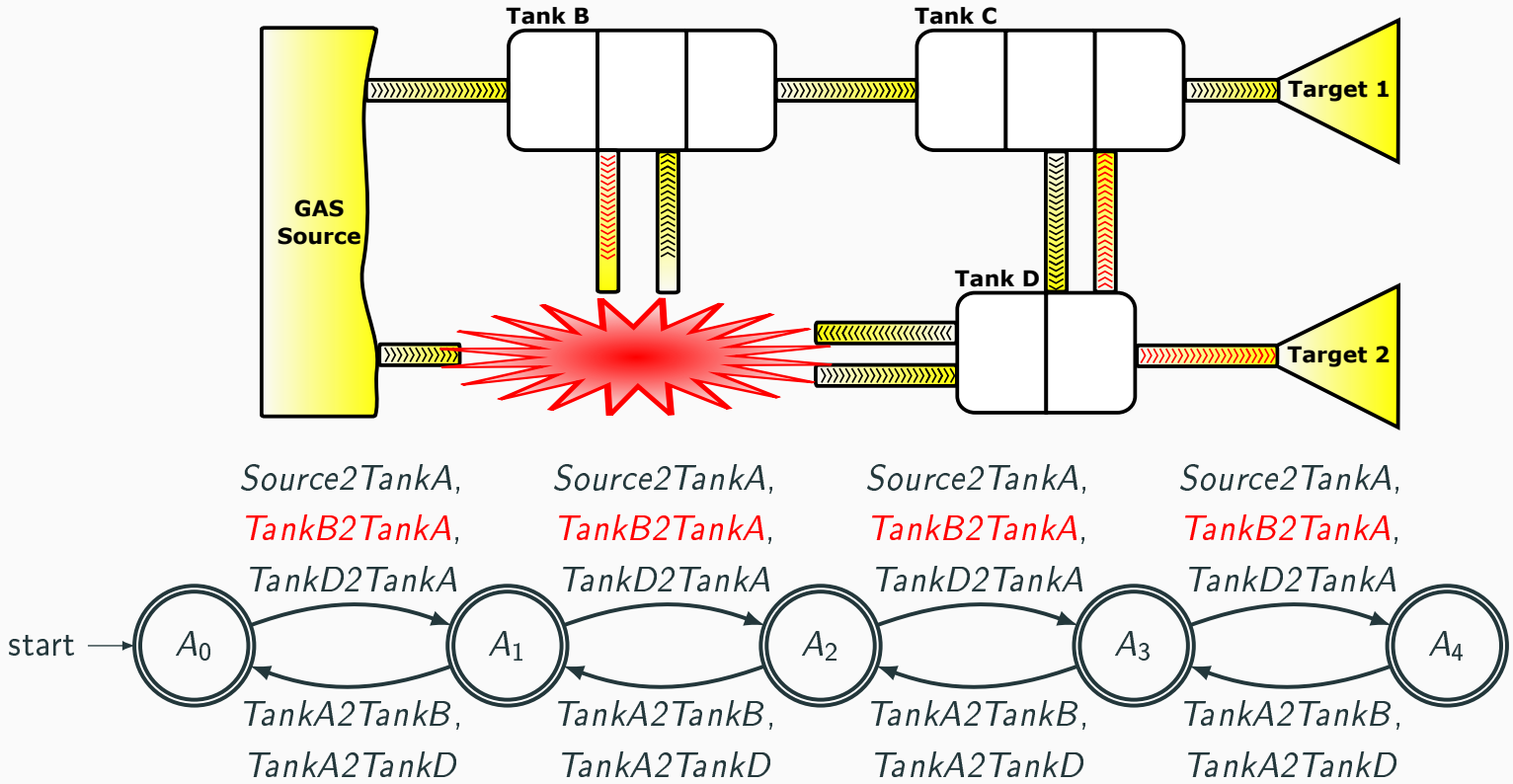
Requirement 1A: TankA does not blow up



Step 2: Detect and remove the illegal state A_B.

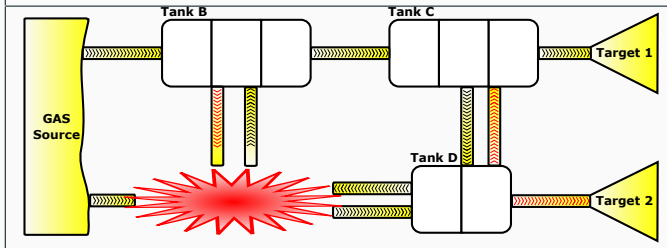
Requirement 1A

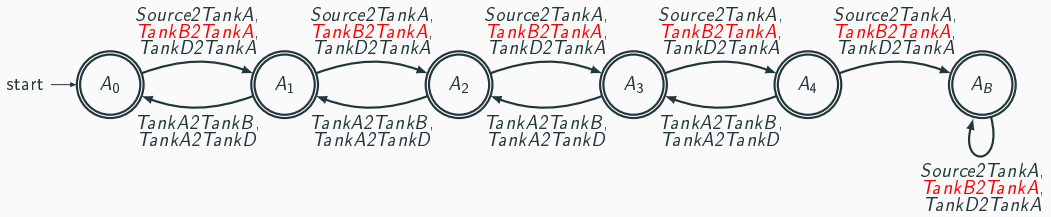
Requirement 1A: TankA does not blow up



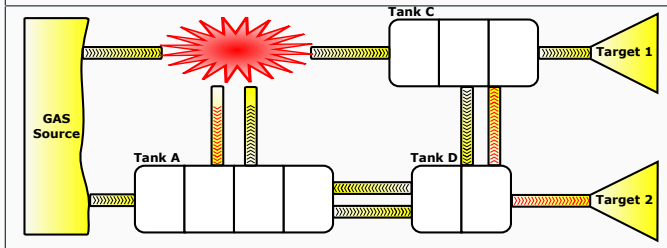
Final requirement.

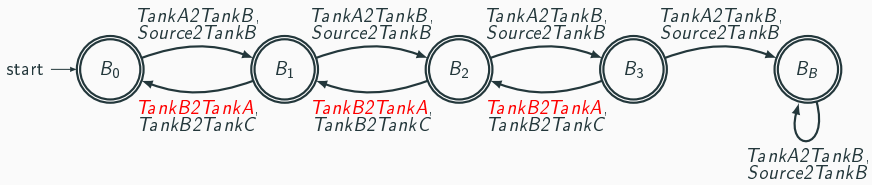
Requirements 1A, 1B, 1C, and 1D - Formalization



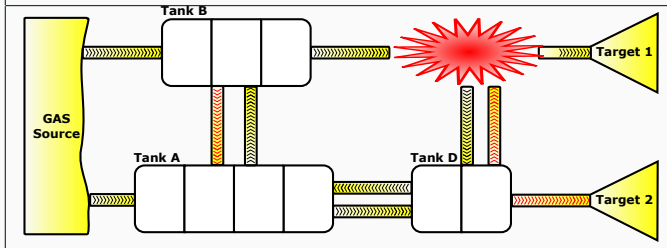


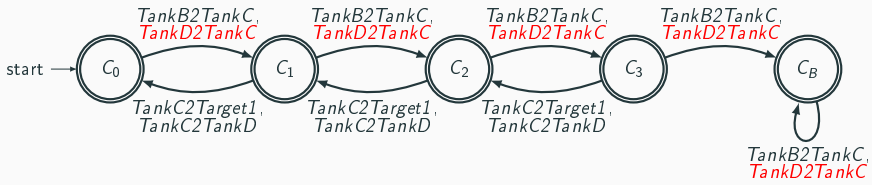
R_{1B}: **TankB does not blow up.**



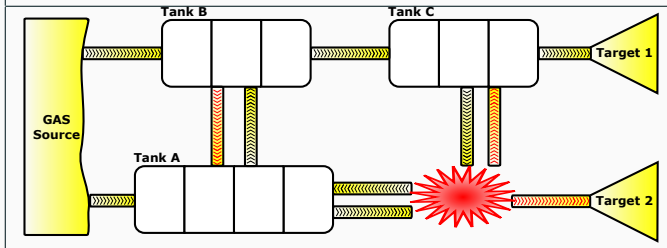


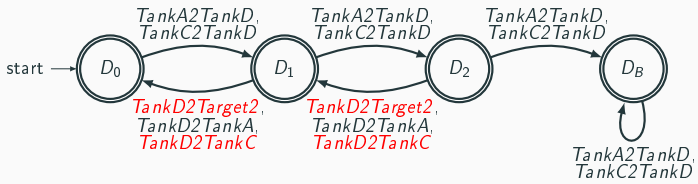
R_{1C}: **TankC does not blow up.**





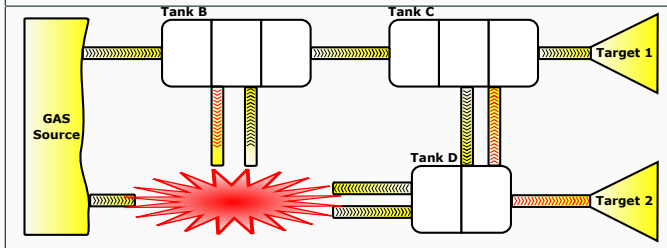
R_{1D}: **TankD does not blow up.**

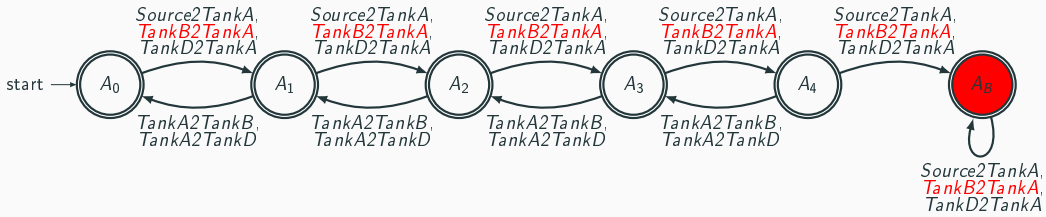




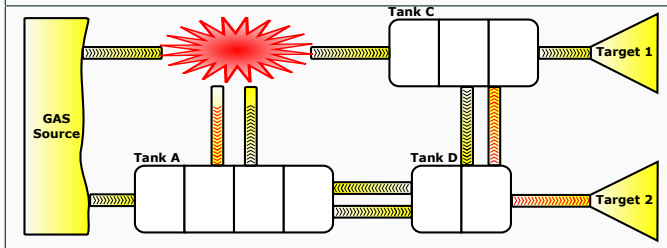
Step 1: Start with marked copies of TankA, TankB, TankC, and TankD

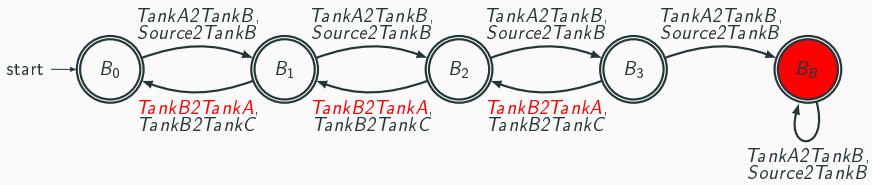
Requirements 1A, 1B, 1C, and 1D - Formalization



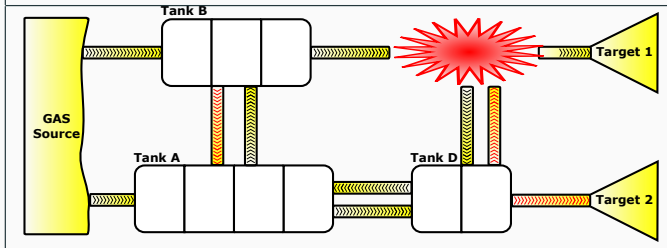


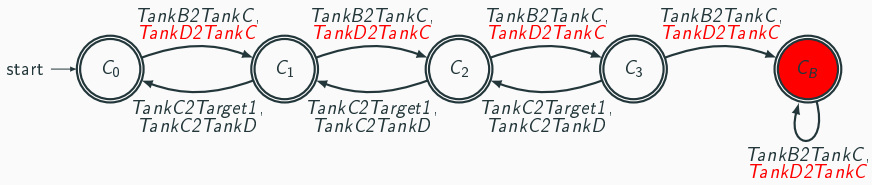
R_{1B}: **TankB does not blow up.**



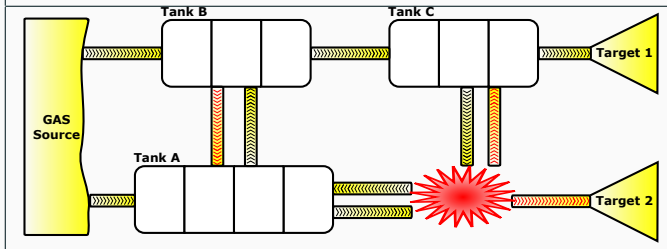


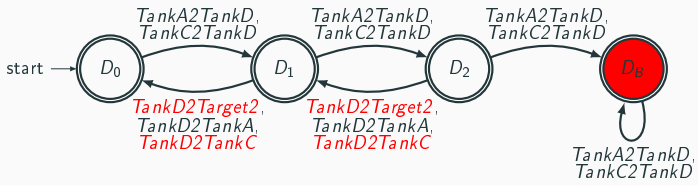
R_{1C}: **TankC does not blow up.**





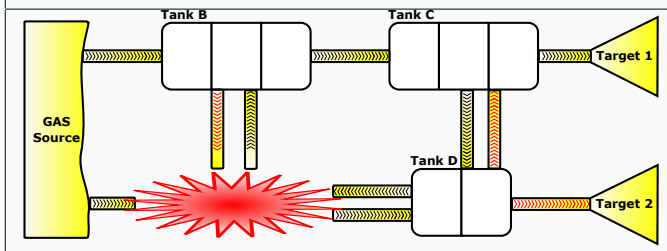
R_{1D}: **TankD does not blow up.**

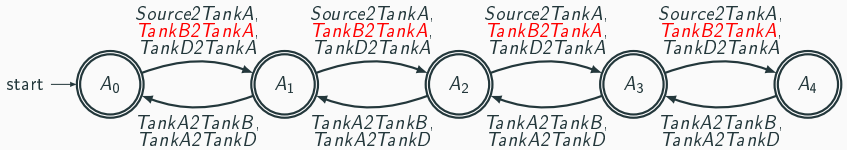




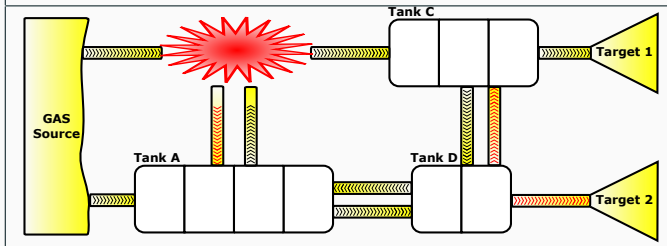
Step 2: Identify and remove the illegal states A_B, B_B, C_B, and D_B.

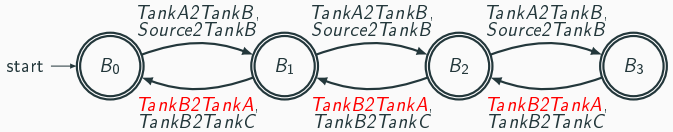
Requirements 1A, 1B, 1C, and 1D - Formalization



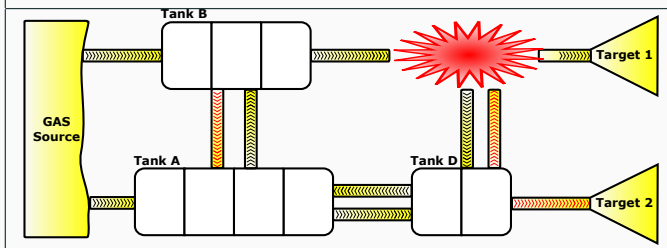


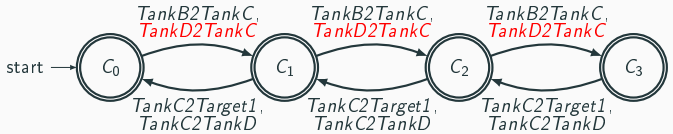
R_{1B}: **TankB does not blow up.**



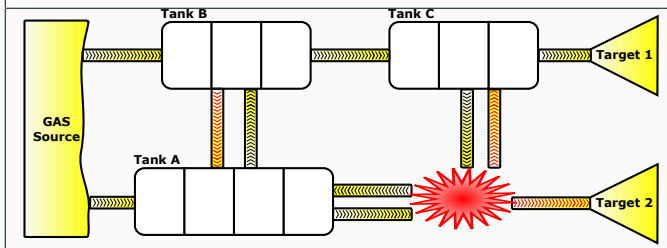


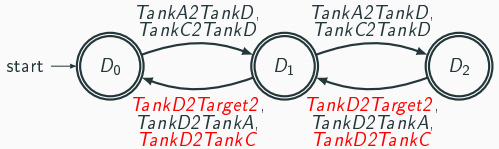
R_{1C}: **TankC does not blow up.**





R_{1D}: **TankD does not blow up.**

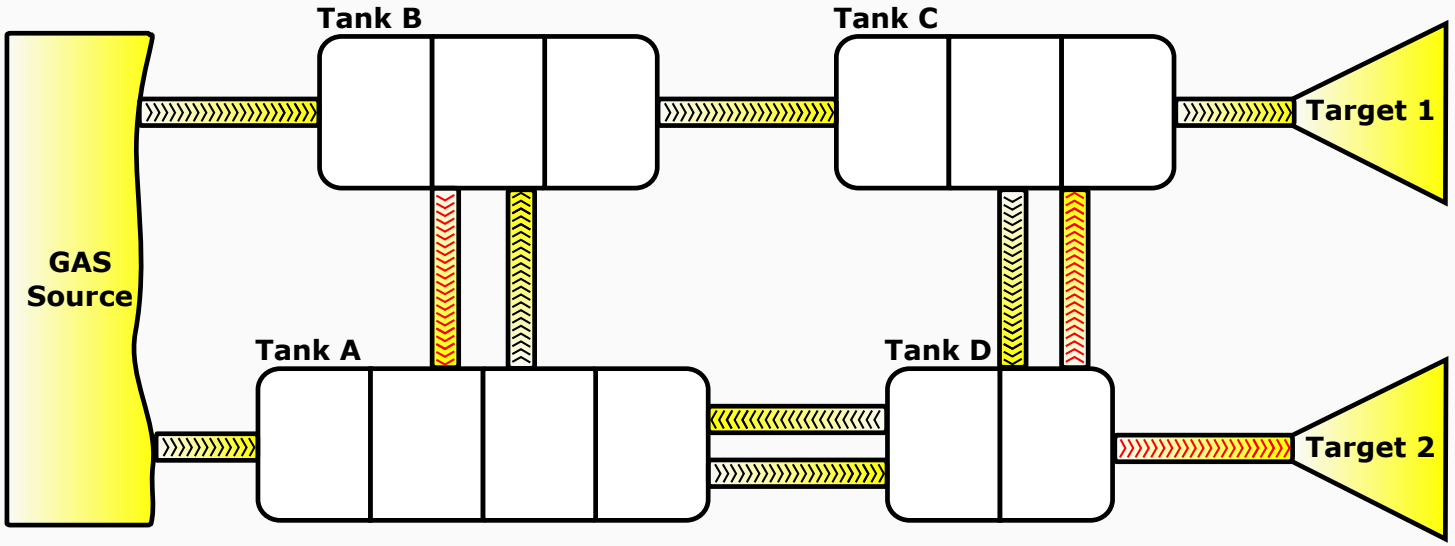




Final sub-requirements. $R_1 := R_{1A} \| R_{1B} \| R_{1C} \| R_{1D}$.

Requirement 2

Requirement 2: The plant must deliver at least 1 gas unit to each Target



How many automata?

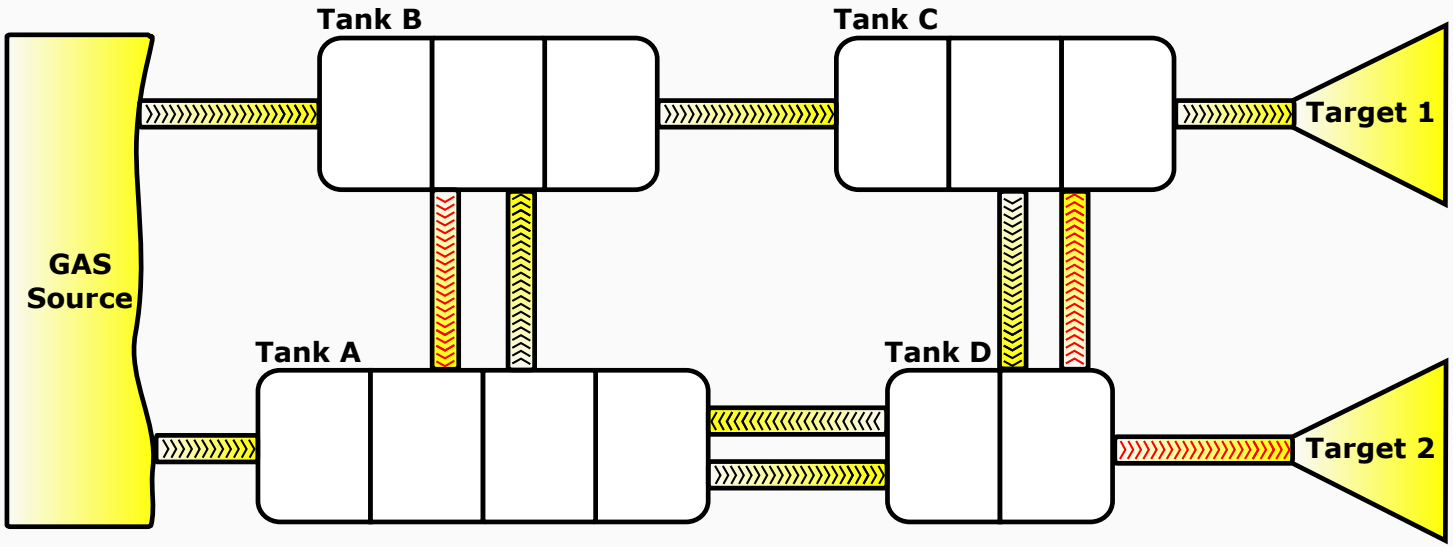
Requirement 2 - Decomposition

Requirement 2: The plant must deliver at least 1 gas unit to each Target

$\Downarrow$

R_{2A} : **The plant must deliver at least 1 gas unit to Target 1**

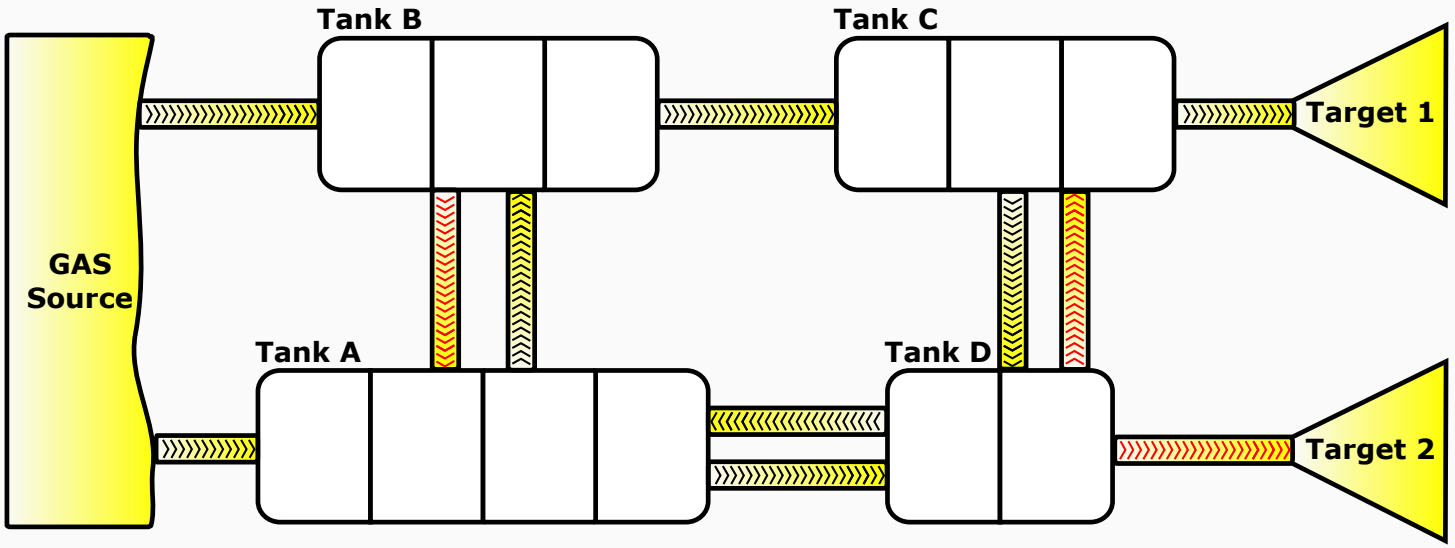
R_{2B} : **The plant must deliver at least 1 gas unit to Target 2**



Hint: some task must be completed

Requirement 2 - Target 1

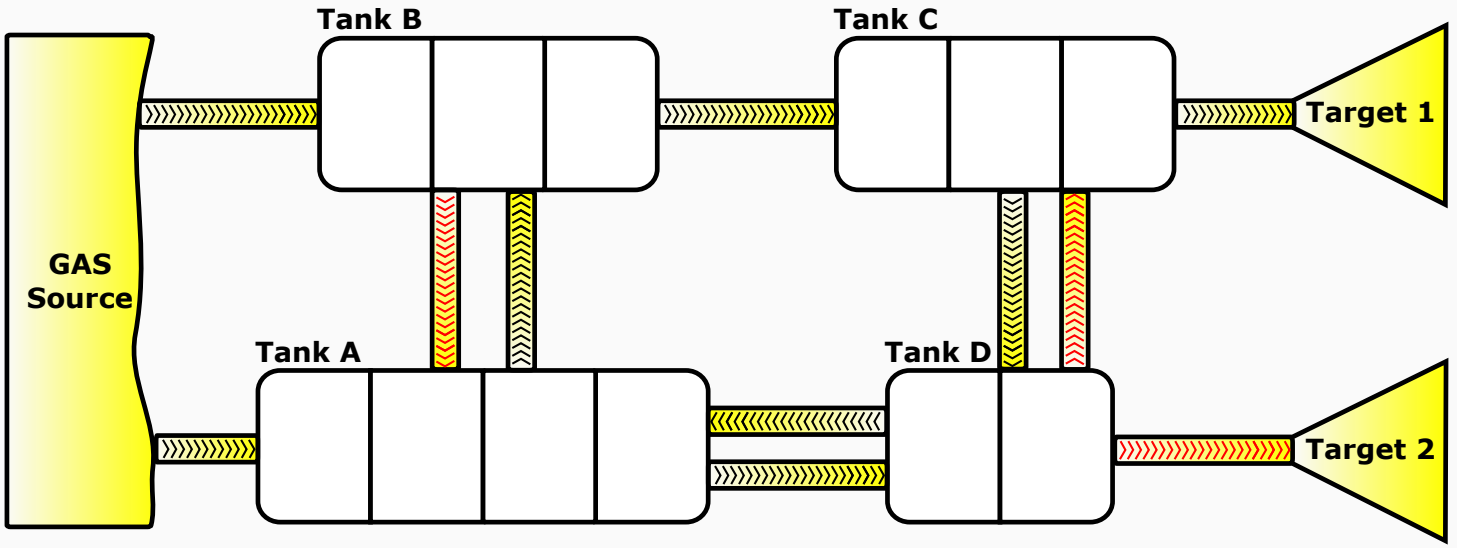
Requirement R_{2A}: The plant must deliver at least 1 gas unit to Target 1

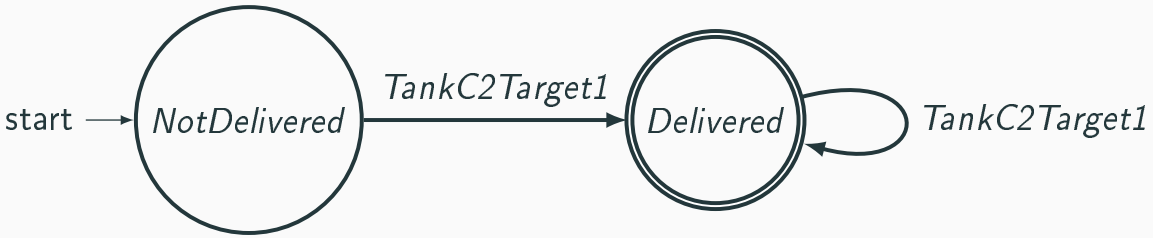


- States?
- **Marking?**
- Transitions?
- Event controllability?

Requirement 2 - Target 1

Requirement R_{2A}: The plant must deliver at least 1 gas unit to Target 1

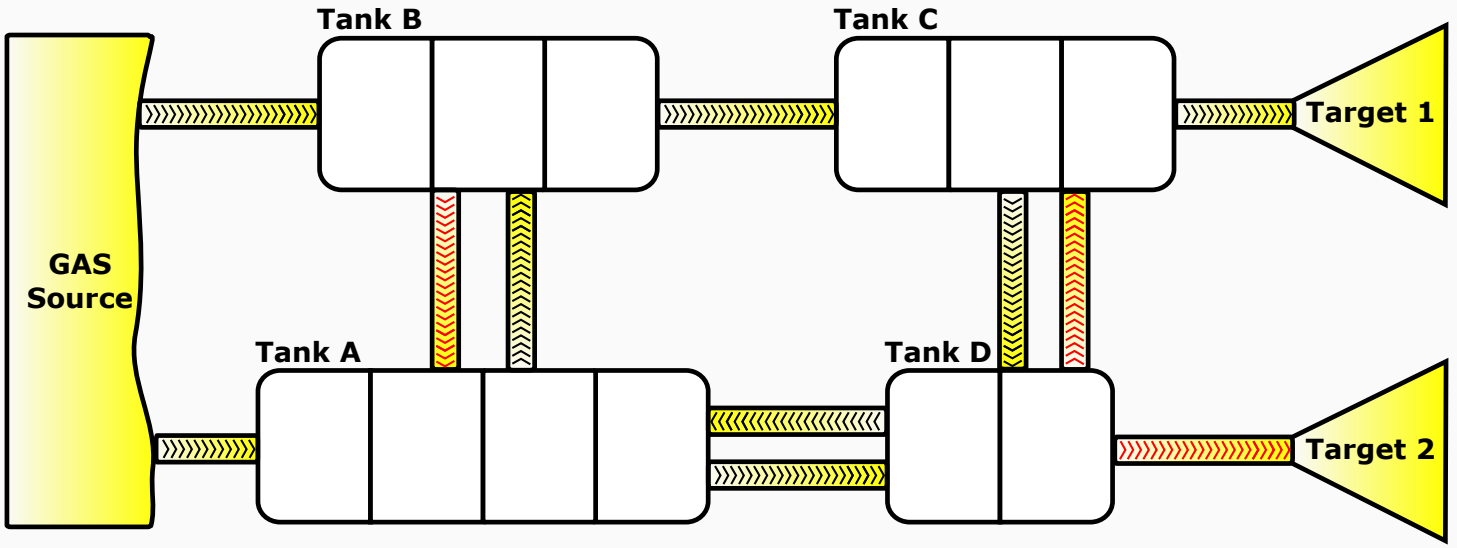


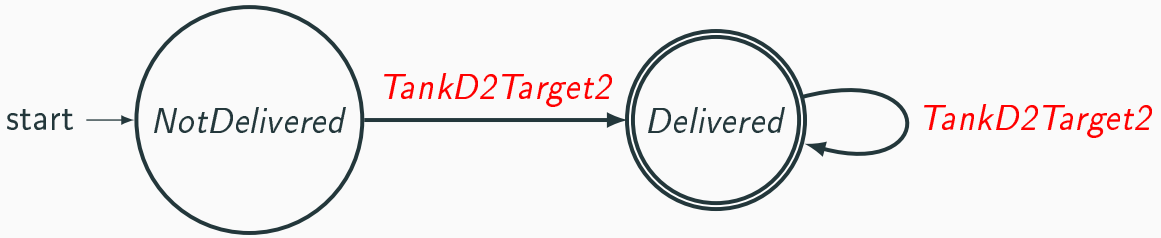


Take a moment to reason on the marking.

Requirement 2 - Target 2

Requirement R_{2B}: The plant must deliver at least 1 gas unit to Target 2

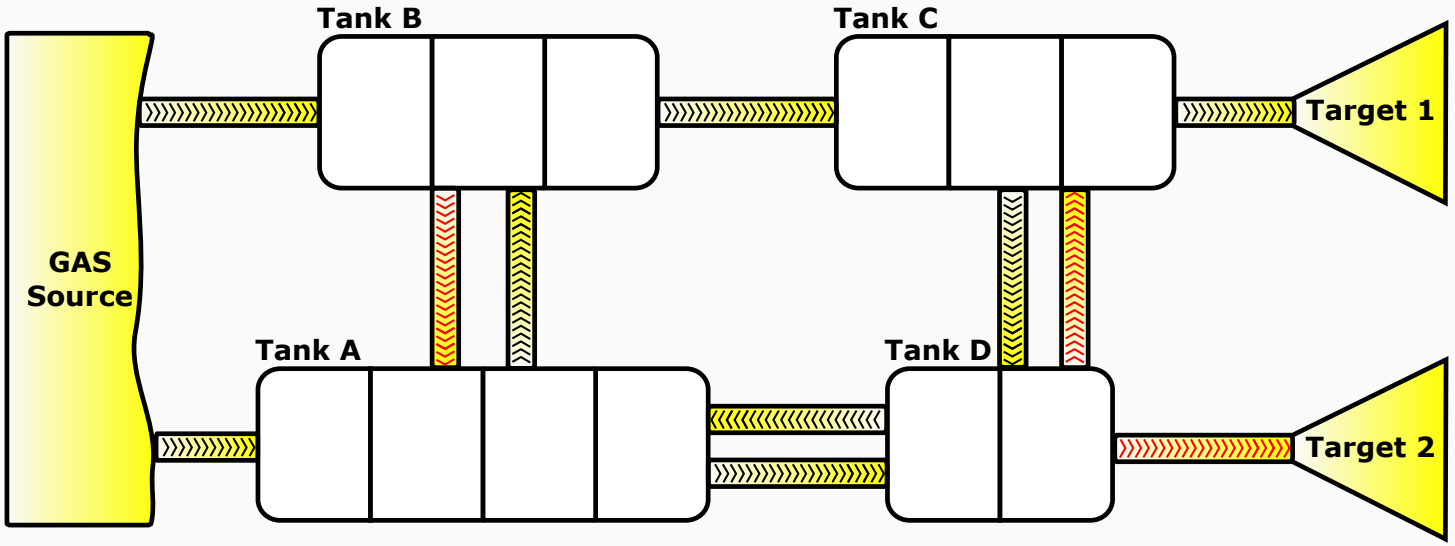




So, $R_2 := R_{2A} \| R_{2B}$

Requirement 3

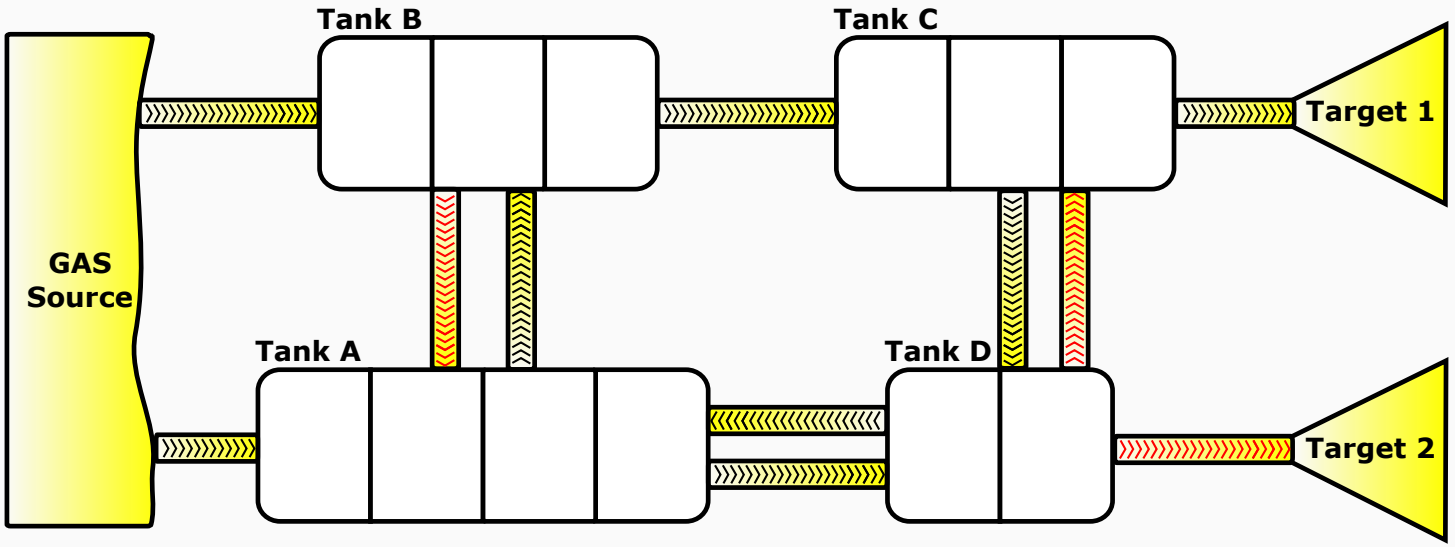
Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate



Straightforward but there is something to pay attention to... what?

Requirement 3

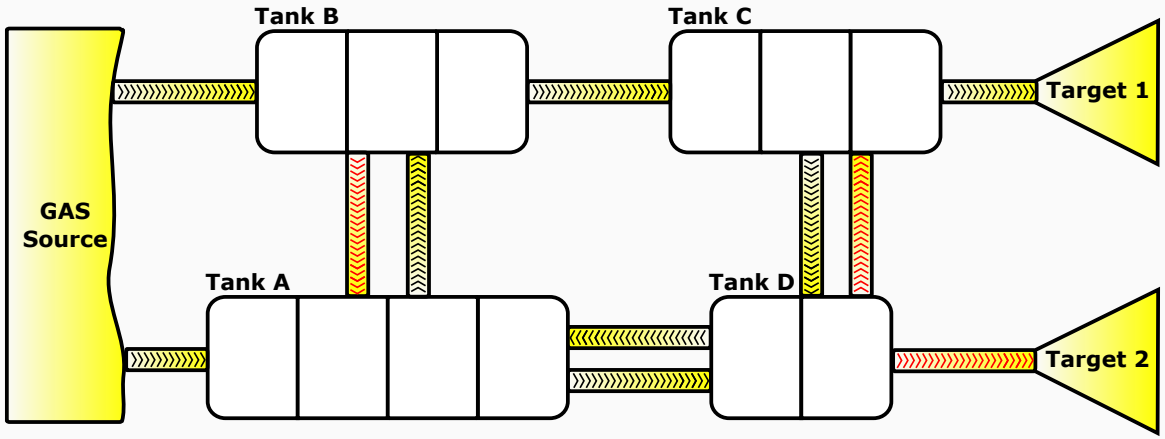
Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate



Which target will be supplied first? That's up to you.. but the requirement doesn't actually impose such a condition...So?

Requirement 3 - Attempt 1 - Non-deterministic

Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate



R_{3A}: **If Target 1 is supplied first...**

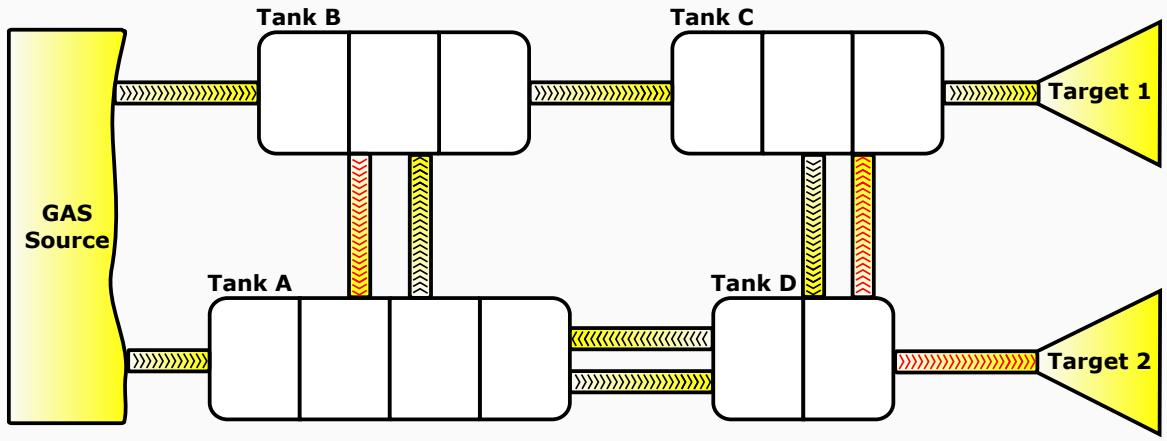
???

R_{3B}: **If Target 2 is supplied first...**

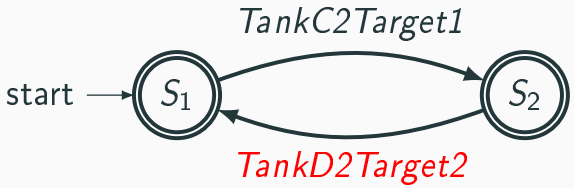
???

Requirement 3 - Attempt 1 - Non-deterministic

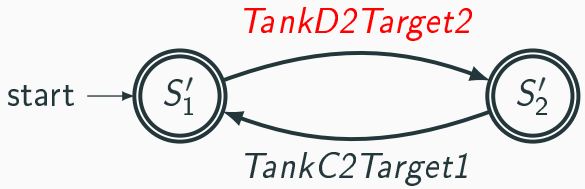
Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate



R_{3A}: **If Target 1 is supplied first...**



R_{3B}: **If Target 2 is supplied first...**

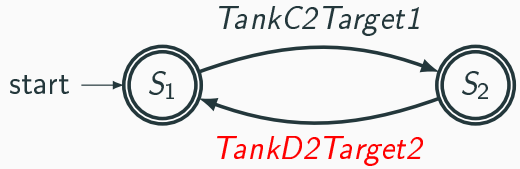
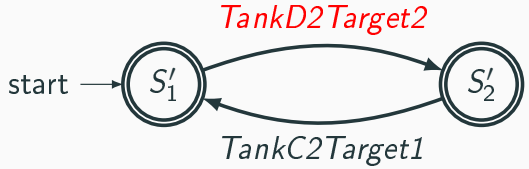
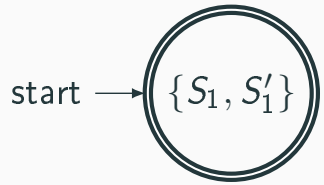
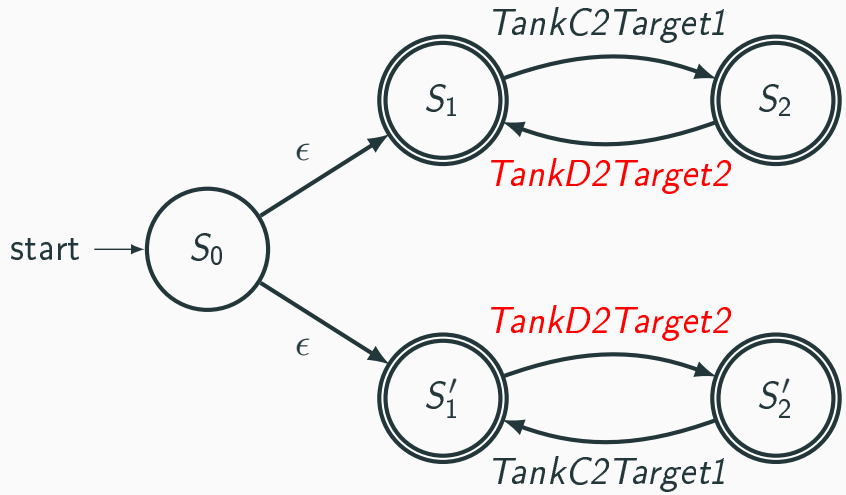


Not certainly an AND of the two automata.

We need the UNION of these two automata!

Requirement 3 - Attempt 1 - Nondeterministic

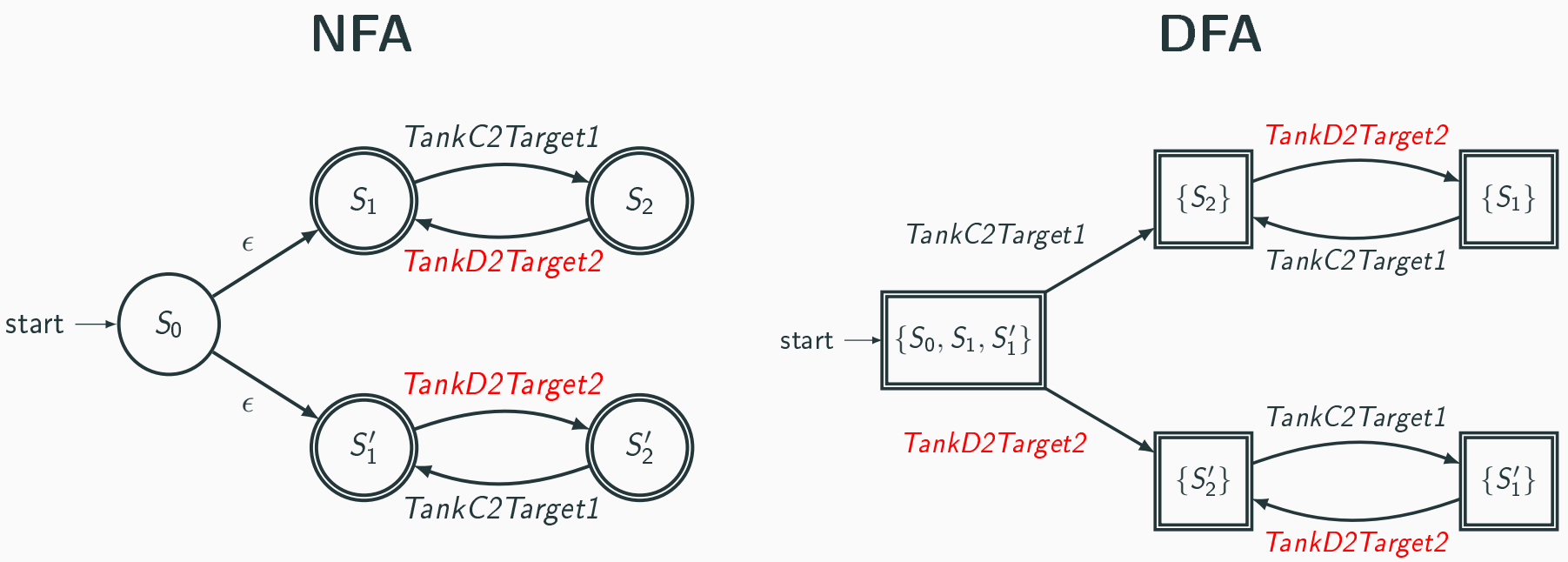
Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate

Requirement R_{3A}	**Requirement R_{3B}**
	
$R_{3A} \wedge R_{3B} := R_{3A} \| R_{3B} = R_{3A} \times R_{3B}$	$R_{3A} \vee R_{3B}$
	

Homework: synthesize a supervisor that (also) takes into consideration requirement $R_{3A} \wedge R_{3B}$. What effect does it have on the plant?

Requirement 3 - Attempt 1 - Nondeterministic

Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate



Requirement 3 - Attempt 2 - Deterministic

Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate

$\Downarrow$

3A) If Target 1 is supplied, then Target 2 must be supplied at least once before Target 1 is supplied again.

3B) If Target 2 is supplied, then Target 1 must be supplied at least once before Target 2 is supplied again.

. . .

. . .

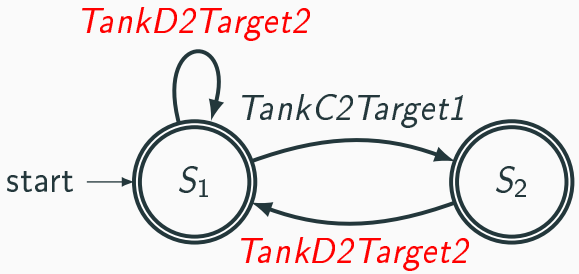
If Target $i = 1, 2$ is supplied, then Target $(i \mod 2) + 1$ must be supplied at least once before Target i is supplied again.

Requirement 3 - Attempt 2 - Deterministic

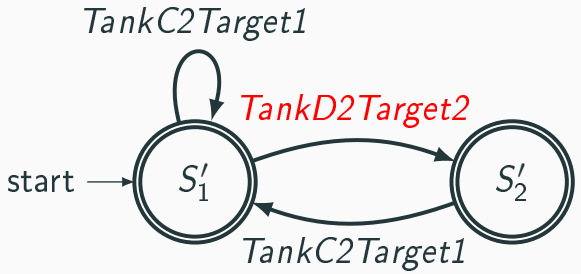
Requirement 3: Gas deliveries to Target 1 and Target 2 must always alternate

$\Downarrow$

3A) If Target 1 is supplied, then Target 2 must be supplied at least once before Target 1 is supplied again.



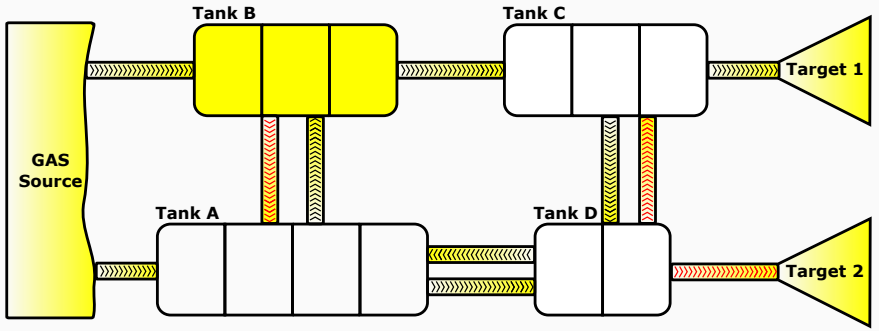
3B) If Target 2 is supplied, then Target 1 must be supplied at least once before Target 2 is supplied again.



If Target $i = 1, 2$ is supplied, then Target $(i \mod 2) + 1$ must be supplied at least once before Target i is supplied again.

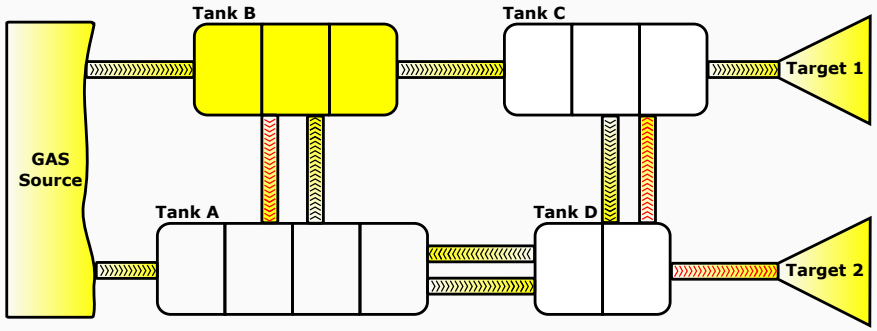
Requirement 4

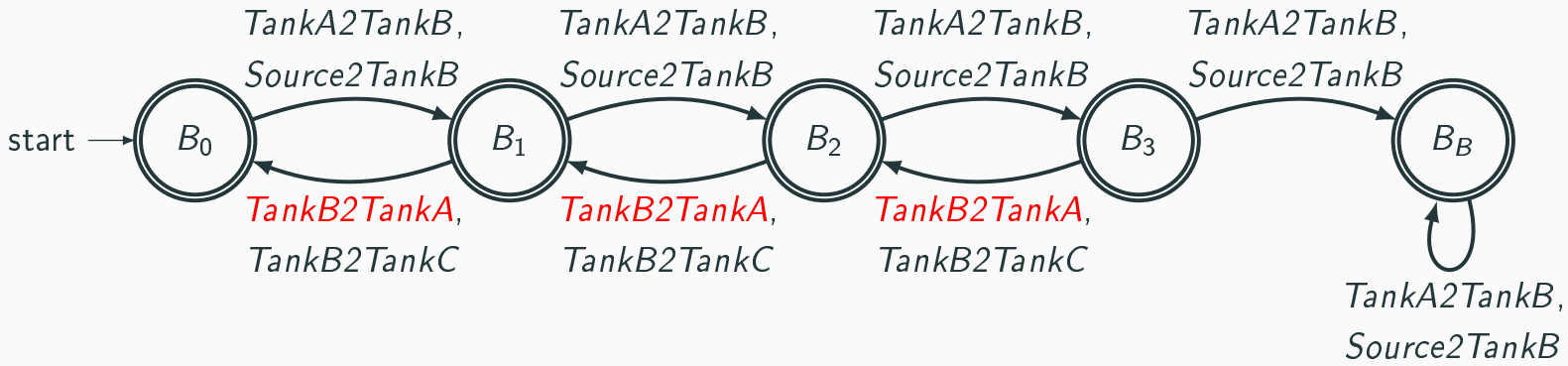
Requirement 4: TankB can supply TankC only if TankB is full



- States?
- Transitions?
- Event controllability?

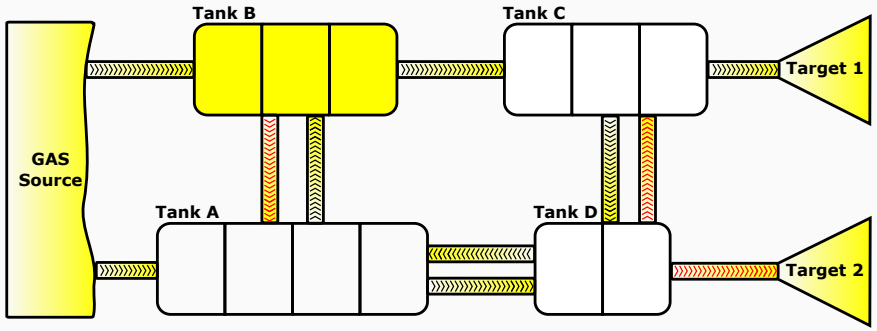
Requirement 4: TankB can supply TankC only if TankB is full

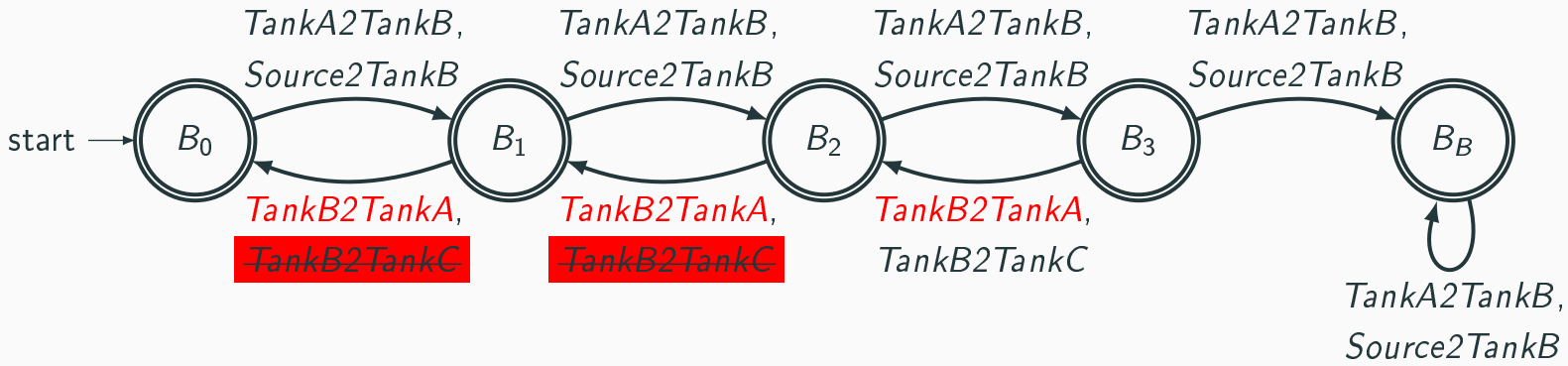




Step 1: Start from a copy of the automaton for Tank B.

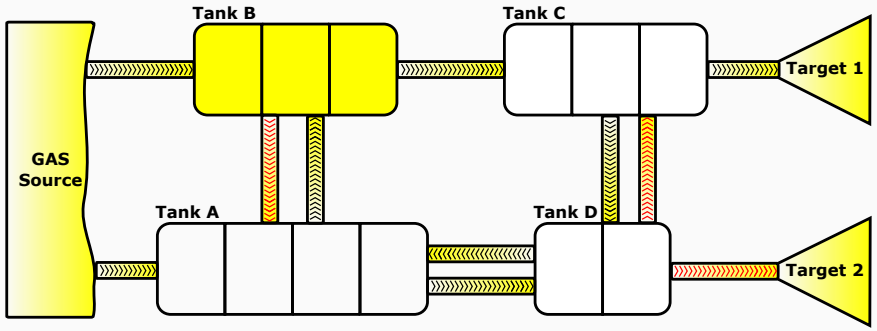
Requirement 4: TankB can supply TankC only if TankB is full

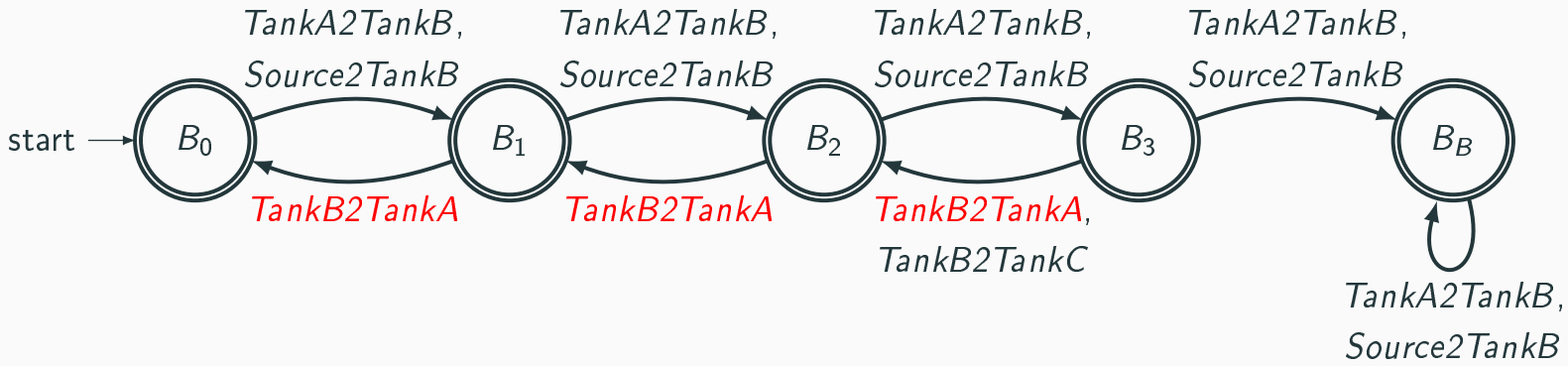




Step 2: Detect and remove the illegal transitions.

Requirement 4: TankB can supply TankC only if TankB is full





Final requirement.