Max Stein
MTKNR: 383687

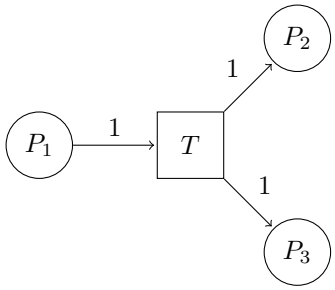
1

1.1

Let the places be numbered by $p_1, \ldots, p_5$ and the transitions also by $t_1, \ldots, t_5$ with $p_1, \ldots, p_3$ for the first row (starting from left and going to right) and p_4, p_5 for the second row and similarly for the transitions with the transitions t_1, t_2 in the first and t_3, t_4, t_5 in the second row. Then $D := \{p_1, p_4\}$ is a trap since $D^* = \{t_1, t_3\}$ and ${}^*D := \{t_1, t_3, t_4\}$, hence $D^* \subseteq {}^*D$. $D := P \backslash \{p_2\}$, with P all the places, is a siphon since even any transition "reduces" at least one place of D, meaning $D^* = T$, with T the set of all transitions. An invariant would be $I := (1, 0, 1, 1, 1)$ since any transition reduces and increases exactly one element from $D := P \backslash \{p_2\}$, or one could also simply verify that $I^T * C = 0$.

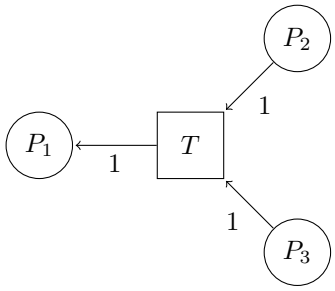
1.2

Traps and siphons are closed under union since for sets of places D_1 and D_2 one has $D_* = D_1^* \cup D_2^*$ and similarly ${}^*D =^* D_1 \cup^* D_2$, so for traps, from $D_1^* \subseteq^* D_1$ and $D_2^* \subseteq^* D_2$ one retrieves $D_1^* \cup D_2^* \subseteq^* D_1 \cup^* D_2$ and so $D^* \subseteq^*$ and analogously for siphons. They are both not closed under intersection or difference, since for traps



$D_1 := \{P_1, P_2\}$ and $D_2 := \{P_1, P_3\}$ are traps but $D_1 \cap D_2 = P_1$ is not as well as $D_1 := \{P_1, P_2\}$ and $D_2 := \{P_2\}$ are traps but $D_1 \setminus D_2 = P_1$ is not.

And analogously for siphons



$D_1 := \{P_1, P_2\}$ and $D_2 := \{P_1, P_3\}$ are siphons but $D_1 \cap D_2 = P_1$ is not as well as $D_1 := \{P_1, P_2\}$ and $D_2 := \{P_2\}$ are traps but $D_1 \setminus D_2 = P_1$ is not.

2

2.1

They are both monotonous since firing a transaction t yields for a place s the new marking

(1) $M(s) - W(s,t) + W(t,s)$ if $s \notin R(t)$

(2) $W(t,s)$ if $R(t) = \left(s, s^{'}\right)$

(3) $M(s) - W(s,t) + (M(s_1) - W(s_1,t)) + W(t,s)$ if $R(t) = \left(s^{'}, s\right)$

which holds for reset and transfer nets, so for two Markings $M, M^{'}$ with $M \leq M^{'}$ one gets in the above always bigger values for $M^{'}$ since the weight values $W(s,t)$, $W(t,s)$ are the same (no matter if one has the Marking M or $M^{'}$) and Marking values like $M(s)$ or $M(s_1)$ are added and not subtracted (meaning that a bigger Marking $M^{'}$ will produce a bigger new Marking).

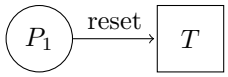
2.2

Transfer nets are strictly monotonous since if $M^{'} > M$ so there is a place s such that $M^{'}(s) > M(s)$ then for any transition t one has

(1) if $s \notin R(t)$ or $(t) = \left(s^{'}, s\right)$ then $M^{'}(s) > M(s)$ after firing t

(2) if $R(t) = \left(s, s^{'}\right)$ then $M^{'}(s^{'}) > M(s^{'})$ after firing t

Reset Nets are not strictly monotonous since e.g. for



no matter of the Initial Marking like $M(P_1) := 0$ or $M(P_1) := 1 \ldots$ one always gets after firing the empty state $M = 0$.

2.3

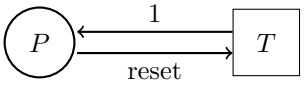
The TerminationCheck algorithm works for both nets since it will certainly terminate (else one would get an infinite path of Markings $M_0, M_1, \ldots$ where $\exists i, j,\ i < j : M_i \leq M_j$ meaning that the algorithm must have stopped after computing M_j which is a contradiction) and if it terminates with TERMINATING then any enabled sequence of transitions must have length smaller than the size of the computed set of pairs E, so no matter of if it's a transition or

reset net there won't be an infinite enabled sequence. If it terminates with NON-TERMINATING then the algorithm has found two markings $M, M^{'}$ with $M \leq M^{'}$ and since transition and reset nets are both monotonous one can always apply the transition sequence from M to $M^{'}$ and it will always stay enabled, hence the net is non-terminating.

2.4

The BoundednessCheck algorithm works only for transition nets since it will for a similar reason as in 2.3 terminate and if it returns TERMINATING then there are only finitely many markings reachable, so the net is bounded in this case. If it returns non-terminating then the algorithm has found two markings $M, M^{'}$ with $M < M^{'}$ and since transition nets are strictly monotonous one can apply infinitely often the sequence from M to $M^{'}$ and the marking will always get strictly bigger, which means that for a place s with $M^{'}(s) > M(s)$ the number of tokens is unbounded.

For the case of reset nets the algorithm will for the following net

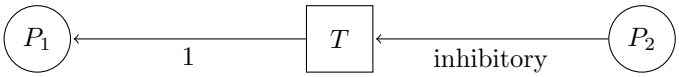


with starting marking $M(P) = 0$ apply the transition T, compute from that the marking $M^{'}$ with $M^{'}(P) = 1$ and then since it it strictly bigger than M it will return NON-TERMINATING, but since for any firing of T the tokens of P are always deleted before one token is added one will forever stay at marking $M^{'}$, hence the net is bounded.

3.

3.1

For the following net the Marking $M^{'}$ with $M^{'}(P_1) = 0; M^{'}(P_2) = 1$ which is bigger than the marking $M = 0$ can't fire the Transition T but M can, so inhibitory edges are not monotonic.



3.2

Given a minsky machine one constructs a petri net by defining for any position in the program (at any command line) a place and for any register a place. One needs for the command $JE(r_i, r_j, k)$ more than one place and one needs one place additionally for the registers which will be used as temporary memory for executing the command $JE(r_i, r_j, k)$. For the commands one defines the

following corresponding transactions where s denotes the corresponding place of the command and the places for the registers are named in exactly the same way as the registers, so r will also denote the place corresponding to register r.

(1) $INC(r)$**:**

$$W(s,t);\ W(t,s^{'});\ W(t,r)$$

where $s^{'}$ denotes the place corresponding to the next command and all the weights above are by default equal to 1 (This works analogously for the decrement command, just by swapping r and t in the third transition)

(2) $JZ(r,z)$**:**

Two transactions t_1, t_2 are created, where the first represents the case that the value of r is zero.

$$W(s,t_1);\ W(t_1,z); T(t_1) = r$$

The second transition makes is possible to continue with the next command in sequence $s^{'}$ if the value in r is bigger than zero

$$W(s,t_2);\ W(t_2,s^{'});\ W(r,t_2);\ W(t_2,r)$$

(3) $JE(r_i,r_j,z)$**:**

Let r denote the additional place for the registers introduced above, then firstly the values of r_i and r_j will be decremented by incrementing the value of r via the transaction t_1

$$W(s,t_1);\ W(t_1,s);\ W(r_i,t_1);\ W(r_j,t_1);\ W(t_1,r)$$

then secondly when r_i or r_j reaches value zero one goes into the states (which all get now defined) s_1 or s_2 depending on which register is zero by using the transitions t_2 and t_3 (s_1 stands for r_i is zero)

$$W(s,t_2);\ W(t_2,s_1);\ T(t_2) = r_i$$

$$W(s,t_3);\ W(t_3,s_2);\ T(t_2) = r_j$$

For the case that r_i and r_j are both zero the state s_3 is defined to which one gets from s_1 and s_3 via the transitions t_4 and t_5 defined by

$$W(s_1,t_4);\ W(t_4,s_3);\ T(t_4) = r_j$$

$$W(s_2,t_5);\ W(t_5,s_3);\ T(t_5) = r_i$$

For the case that the values of r_i and r_j are not equal the state s_4 is defined together with the transitions t_6 and t_7

$$W(s_1,t_6);\ W(t_6,s_4);\ W(r_j,t_6);\ W(t_6,r_j)$$

$$W(s_2,t_7);\ W(t_7,s_4);\ W(r_i,t_7);\ W(t_7,r_i)$$

In both states states s_3 and s_4 (meaning no matter if the register values did match or not) one needs additionally to copy the number of tokens by which the register values have been decremented back from r to r_i and r_j, which is done by the transitions t_8 and t_9

$$W(s_3, t_8);\ W(t_8, s_3);\ W(r, t_8);\ W(t_8, r_i);\ W(t_8, r_j)$$

$$W(s_4, t_9);\ W(t_9, s_4);\ W(r, t_9);\ W(t_9, r_i);\ W(t_9, r_j)$$

if this is finished and the additional register place r contains no more token, one can if s_3 has a token (meaning the if condition is true) jump into the state z and if s_4 has a token (so if condition is false) go to the next command in sequence which is achieved by the transitions t_{10} and t_{11}

$$W(s_3, t_{10});\ W(t_{10}, z); T(t_{10}) = r$$

$$W(s_4, t_{11});\ W(t_{11}, s^{'}); T(t_{11}) = r$$

The T (-mapping) which defines the inhibitory edges will be by default $\perp$ (if not explicitly defined above)