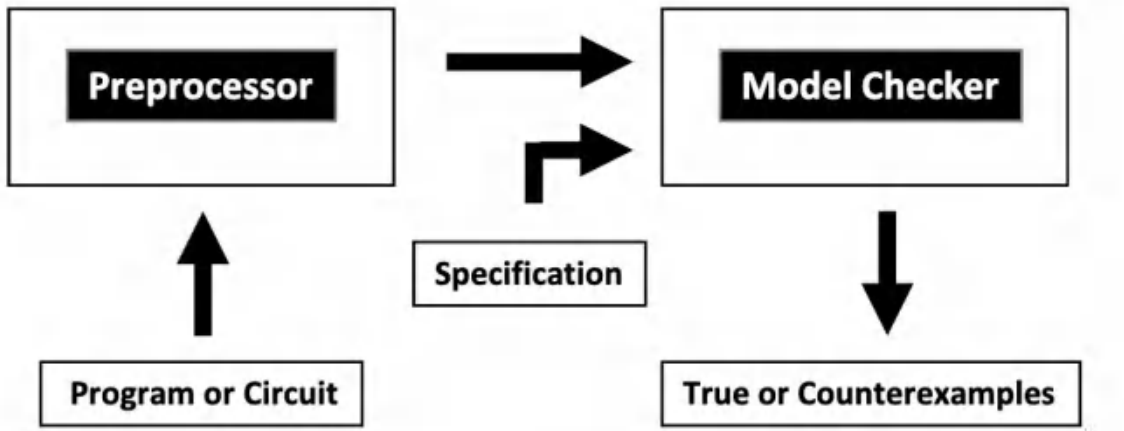
西安电子科技大学
XIDIAN UNIVERSITY

An Innovative Heuristic to Detect Special States in Concurrent Software Systems

Quality and Reliability Engineering International

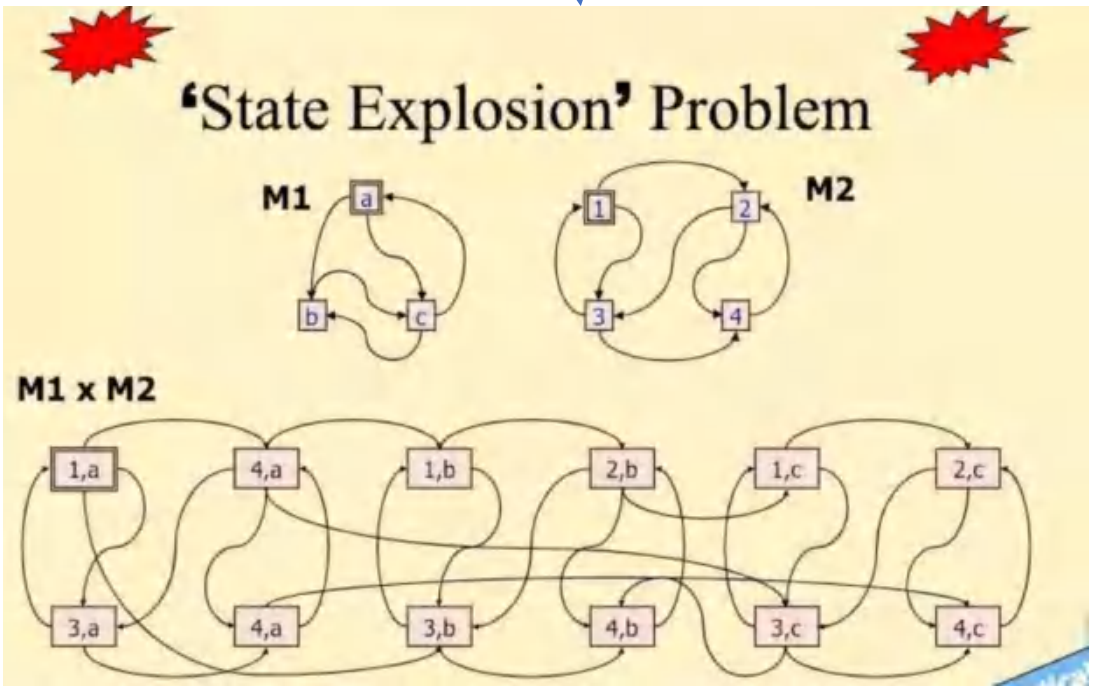
Einollah Pira, Alireza Rouhi

Model Checking and State Explosion

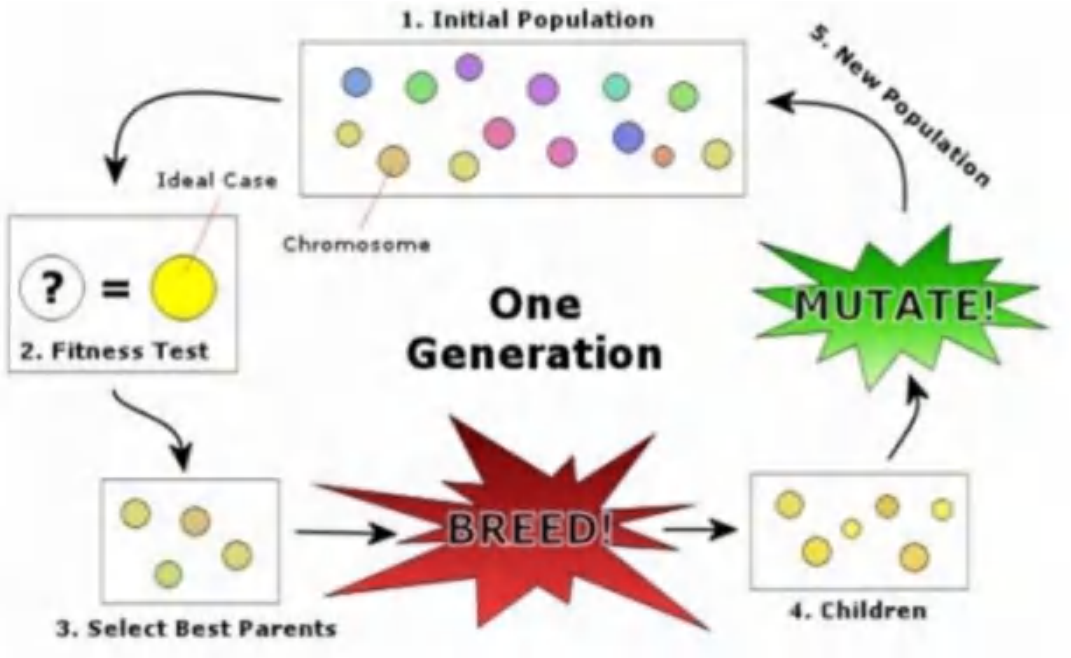


Model checking is a formal verification method used to verify whether a system's model satisfies a set of desired properties

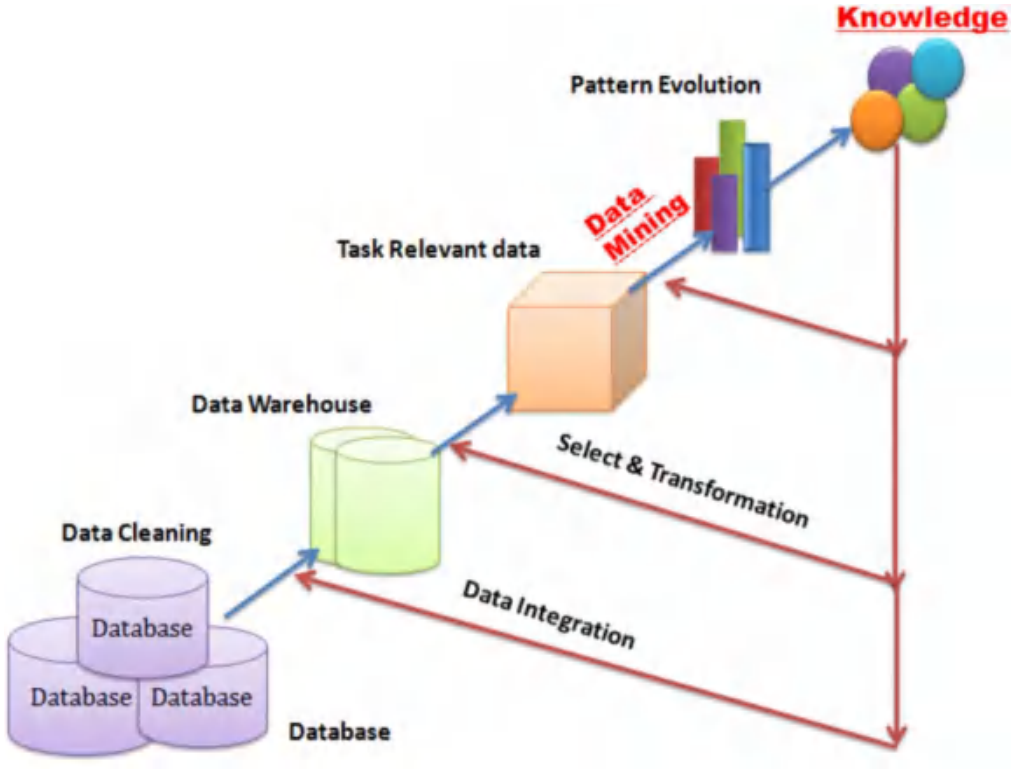
It involves systematically exploring all possible states of the model to check for the presence or absence of specific conditions.



While these methods reduce the number of states to explore, they still suffer from inefficiencies, particularly in large systems, due to the difficulty of estimating the "closeness" to the goal state without exploring too many states



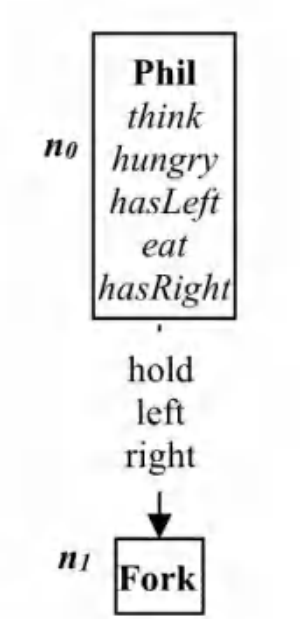
such as behavior graphs for fault detection. While these methods can be effective, they have limitations, especially in systems with infinite state spaces or when suitable smaller models are not available



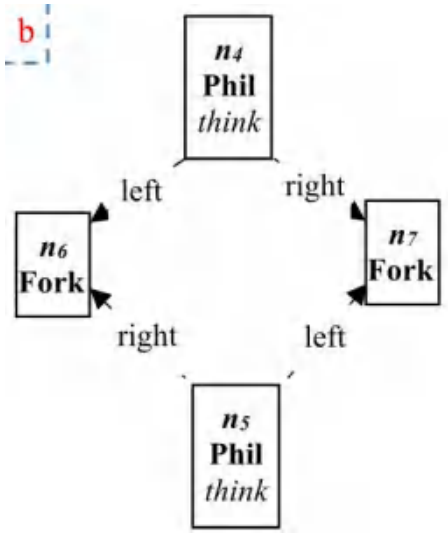
Graph Transformation System (GTS)

GTS

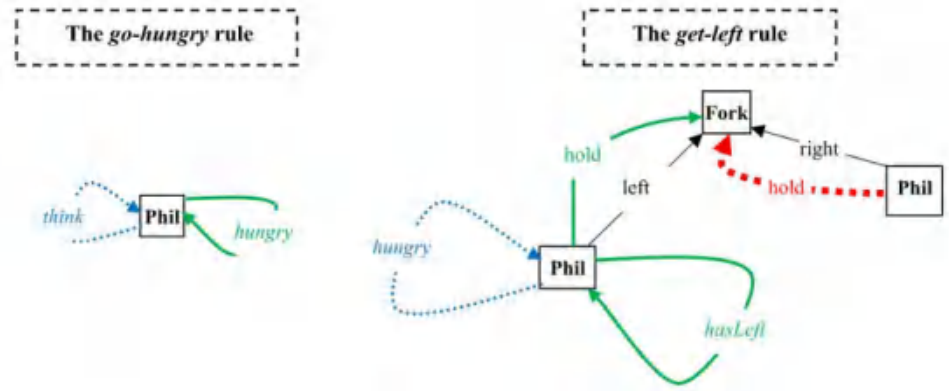
Type Graph



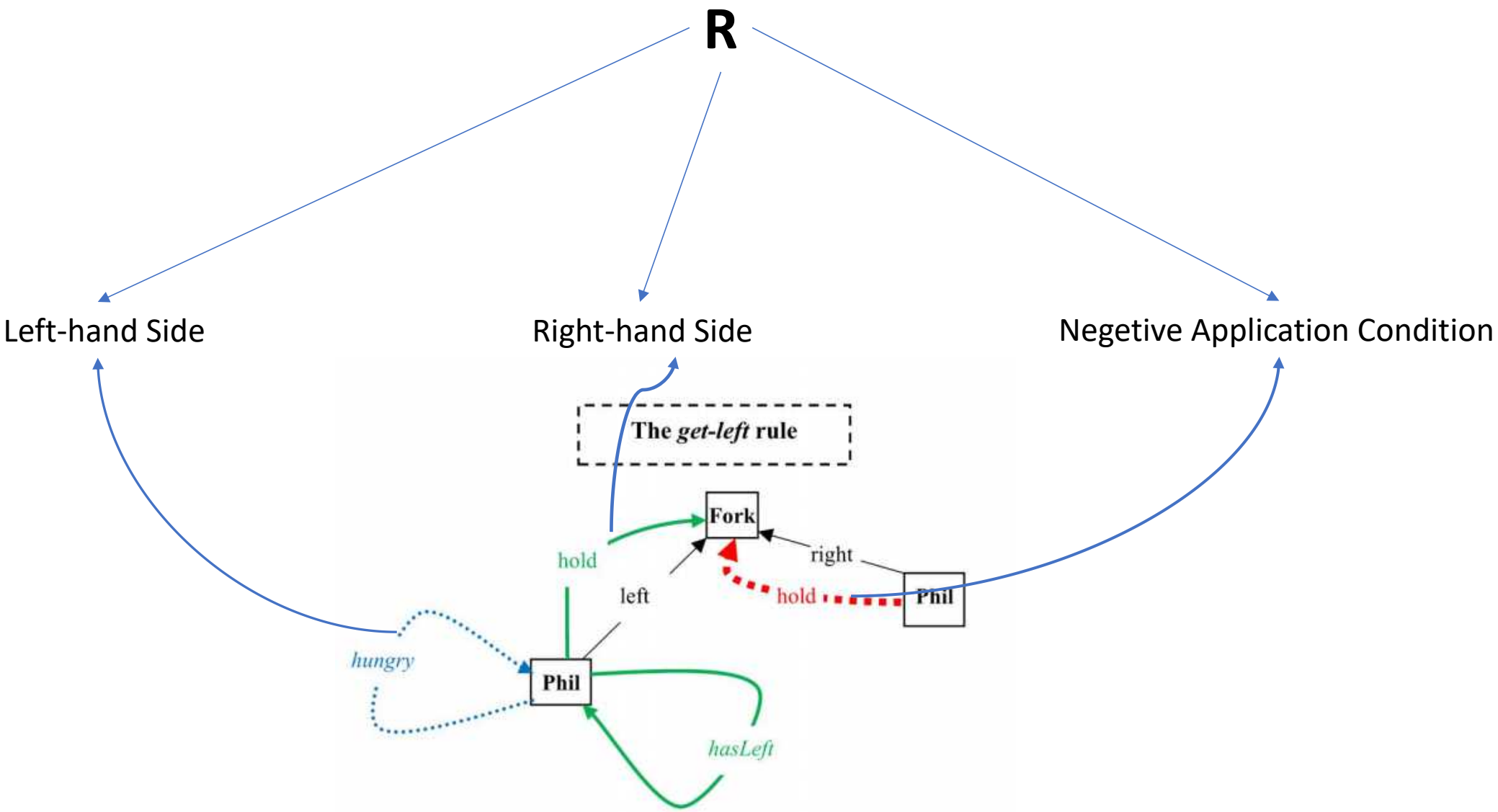
Host Graph



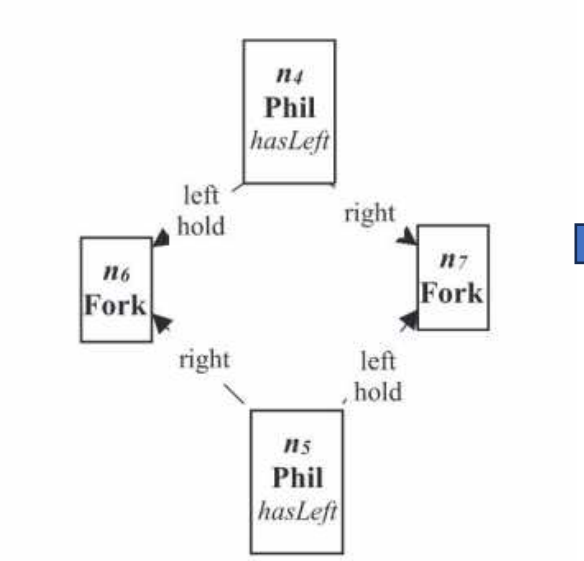
A set of Transformation Rules R



A set of Transformation Rules *R*



Special States in GTS



LHS = {phil+hasLeft+phil, phil+hold+fork}

||

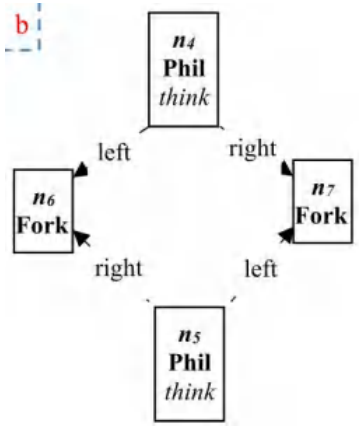
RHS = {phil+hasLeft+phil, phil+hold+fork}

Deadlock State

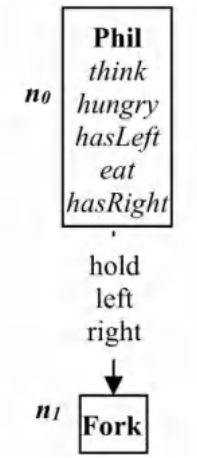
Erase/Create Heuristic（ErCrHeu）

GTS M

M.host



M.Type

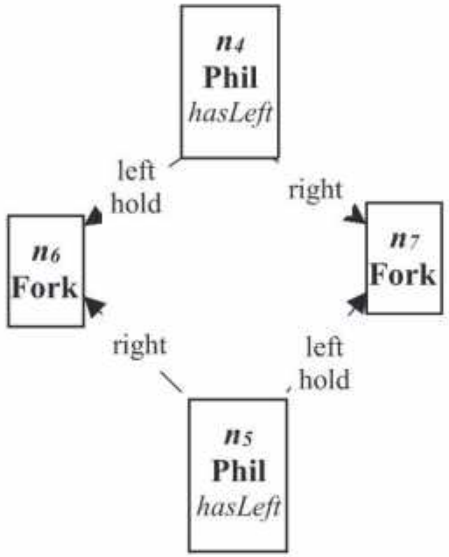


M.Rule = {r1, r2, …, rn}

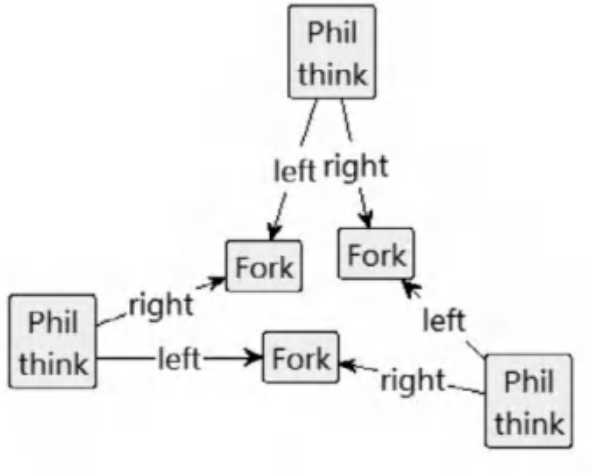
- Rn.LHS
- Rn.RHS
- Rn.NAC

verify/reject

M.Goal

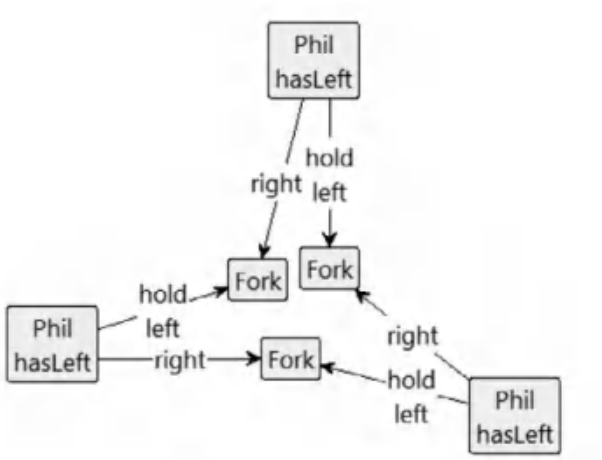


Step of Erase/Create Heuristic



Host graph： ***Start*$_0$**

Start={Phil+think+Phil,
Phil+left+Fork,
Phil+right+Fork}



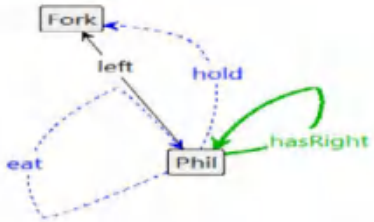
Goal graph： ***Goal***

Goal={Phil+hasLeft+Phil,
Phil+left+Fork,
Phil+hold+Fork,
Phil+right+Fork}

Allowable Rules (named as *AR*) = { }

***Input*: Create an arraylist of *AllStart* with the initial value of *Start*$_0$ for maintaining start sets.**

Transformation Rules

Rule	Graphical view	SRE format
go-hungry		E ={Phil+think+Phil} C ={Phil+hungry+Phil}
get-right		E = {Phil+hasLeft+Phil} C = {Phil+eat+Phil, Phil+hold+Fork}
get-left		E = {Phil+hungry+Phil} C = {Phil+hasLeft+Phil, Phil+hold+Fork}
release-left		E = {Phil+eat+Phil, Phil+hold+Fork} C = {Phil+hasRight+Phil}
release-right		E = {Phil+hasRight+Phil, Phil+hold+Fork} C = {Phil+think+Phil}

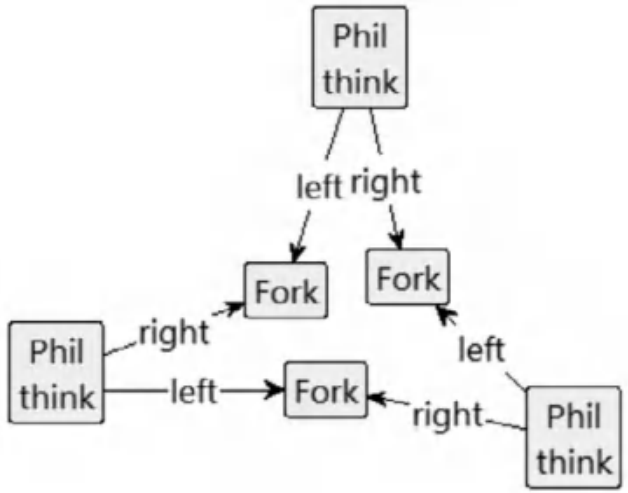
Finding all Start Sets

***Input*: Create an arraylist of *AllStart* with the initial value of $Start_0$ for maintaining start sets.**

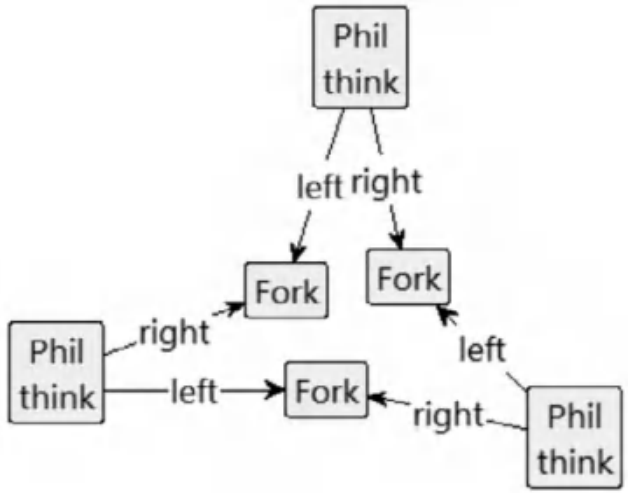
$Start_0$

SRE= {Phil+think+Phil, Phil+left+Fork, Phil+right+Fork}
AR= {*go-hungry*}
prevStartIndex = -1
seqFoundRules ={ *go-hungry* }

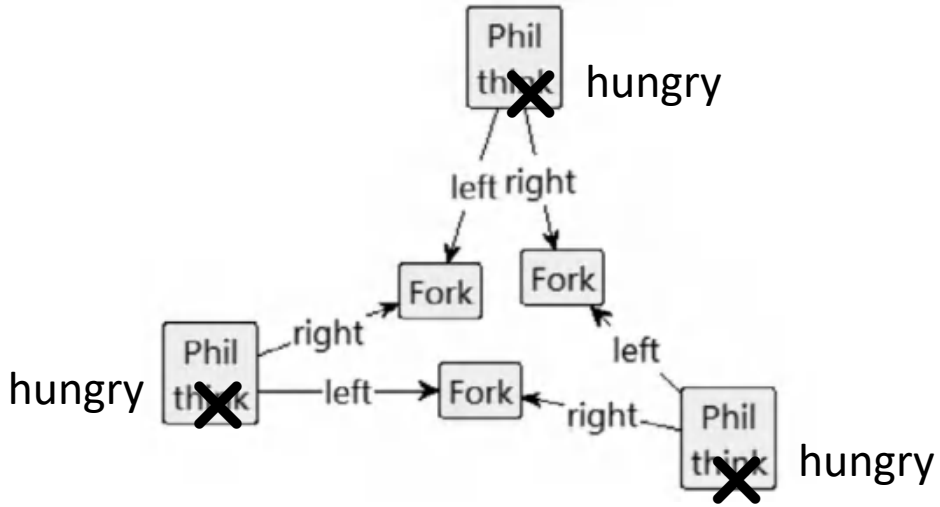
only this rule can be applied on the M.host



Finding all Start Sets



go-hungry



Start$_0$

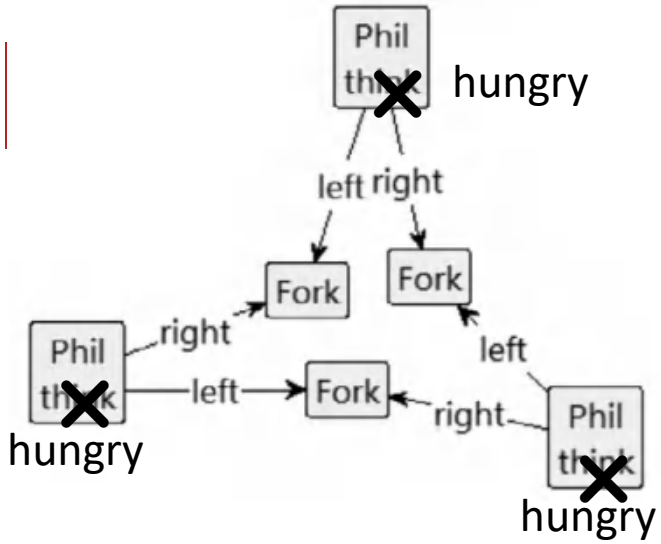
SRE= {Phil+think+Phil, Phil+left+Fork,
Phil+right+Fork}
AR= {*go-hungry*}
prevStartIndex = -1
seqFoundRules ={ *go-hungry* }

go-hungry

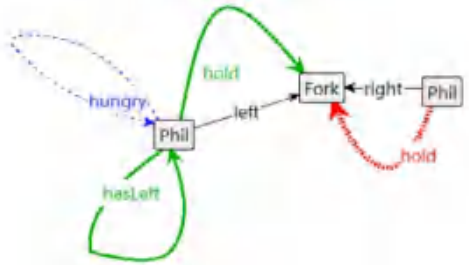
Start$_1$

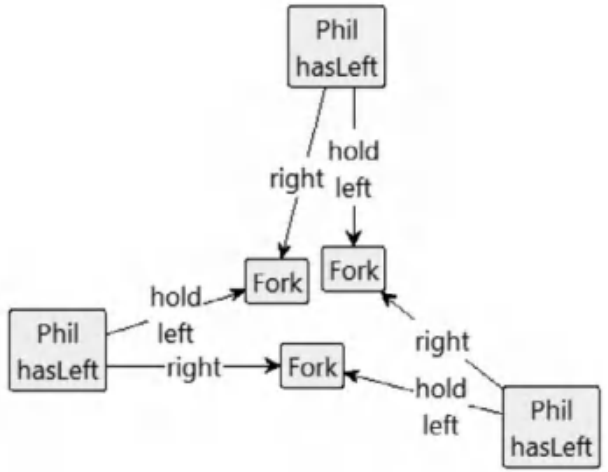
SRE= {Phil+hungry+Phil, Phil+left+Fork,
Phil+right+Fork}
AR = {*get-left*}
prevStartIndex = 0
seqFoundRules ={ *go-hungry* , *get-left* }

Finding all Start Sets



get-left

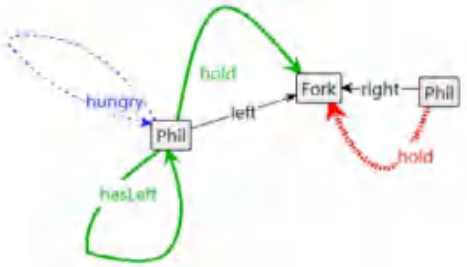




$Start_1$

SRE= {Phil+hungry+Phil, Phil+left+Fork,
Phil+right+Fork}
AR = {*get-left*}
prevStartIndex = 0
seqFoundRules ={ *go-hungry* , *get-left* }

get-left



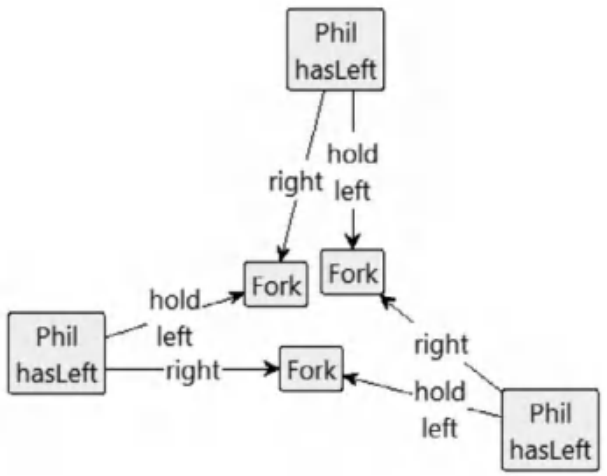
$Start_2$

SRE= {Phil+hasLeft+Phil, Phil+left+Fork,
Phil+right+Fork, Phil+hold+Fork }
AR = {}
prevStartIndex = 1
seqFoundRules ={}

Check and Finish Finding all Start Sets

$$Goal \subseteq Start_2$$

$seqFoundRules = \{ go\text{-}hungry , get\text{-}left \}$



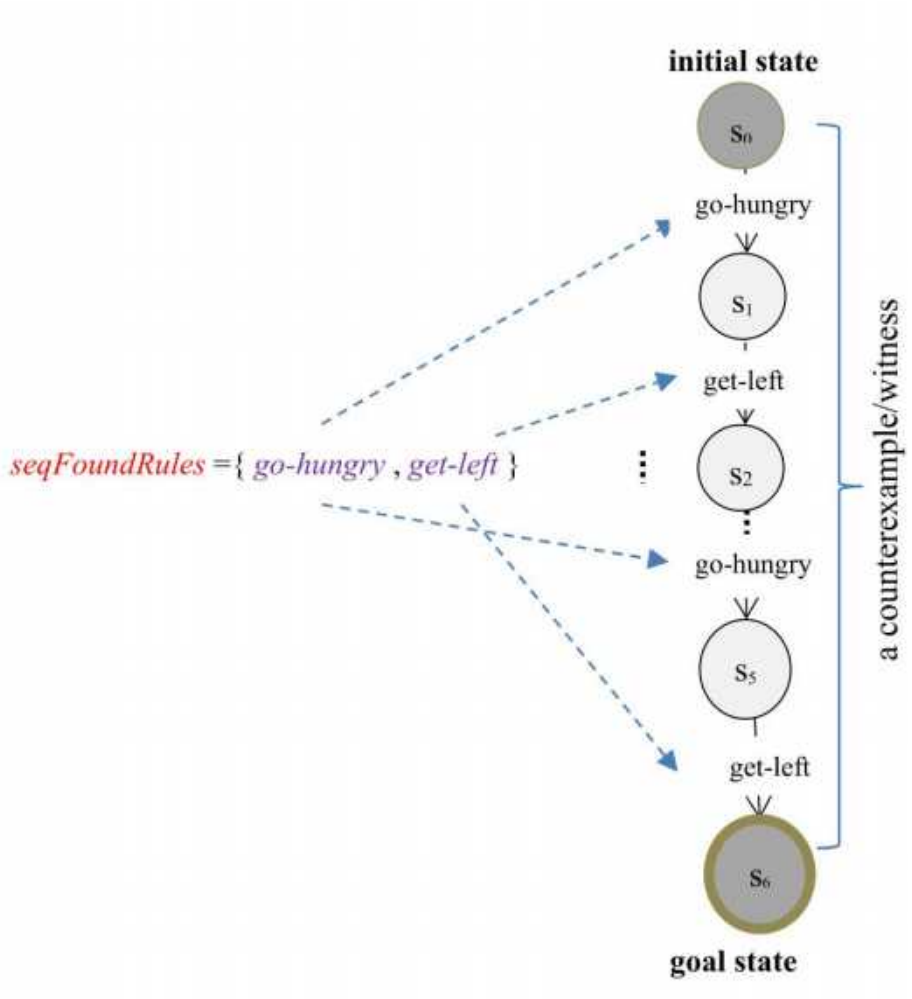
Goal Graph

$\subseteq$

$Start_2$

SRE= {Phil+hasLeft+Phil, Phil+left+Fork,
Phil+right+Fork, Phil+hold+Fork }
$AR = \{\}$
$prevStartIndex = 1$
$seqFoundRules = \{\}$

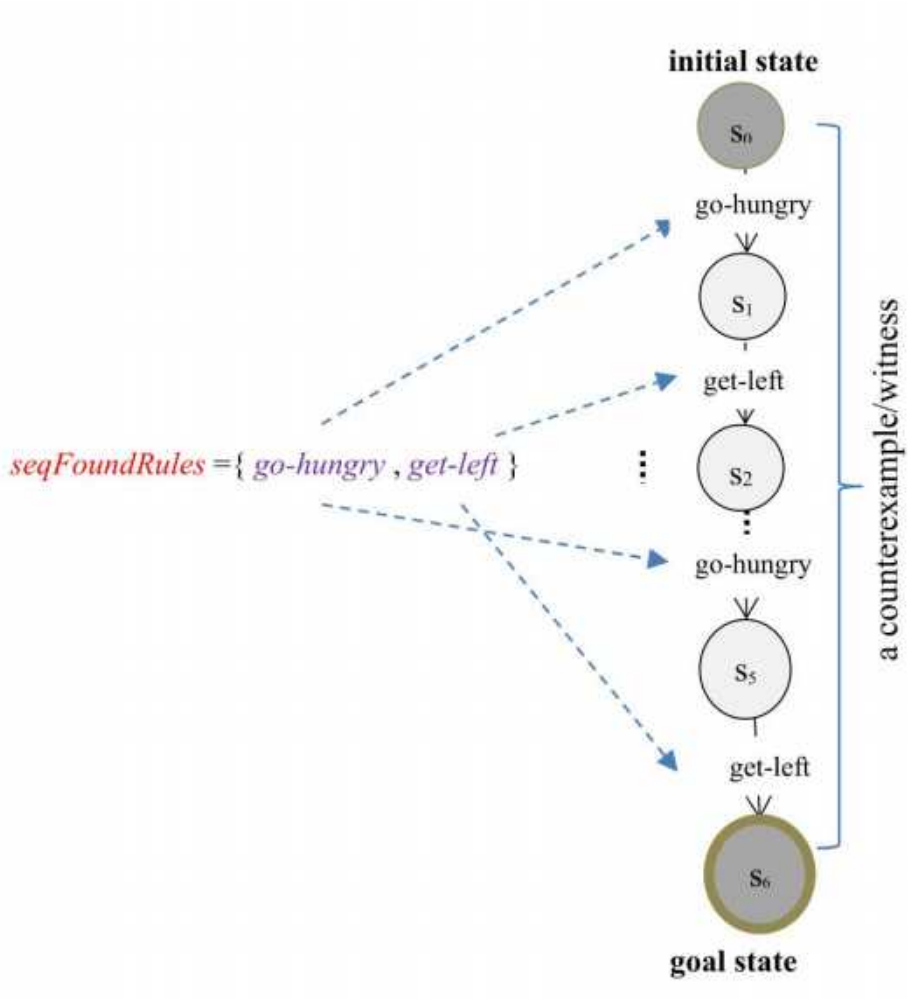
SeqFoundRules for Confirm Result



Each sequence rules comprises a series of rules that have been iteratively encountered along paths leading **to the desired goal state within the state space**.

each sequence rule is employed to explore the model's state space, progressing from the initial state, until either the goal state is attained or the explored path length does not exceed a defined depth (also called **maxDepth**).

SeqFoundRules for Confirm Result



Conversely, if all sequence rules are examined and the goal state is not found, it cannot be concluded that the given property is certainly satisfied/refuted in the model, **because only a small portion of the state space of the model is explored by this heuristic.**