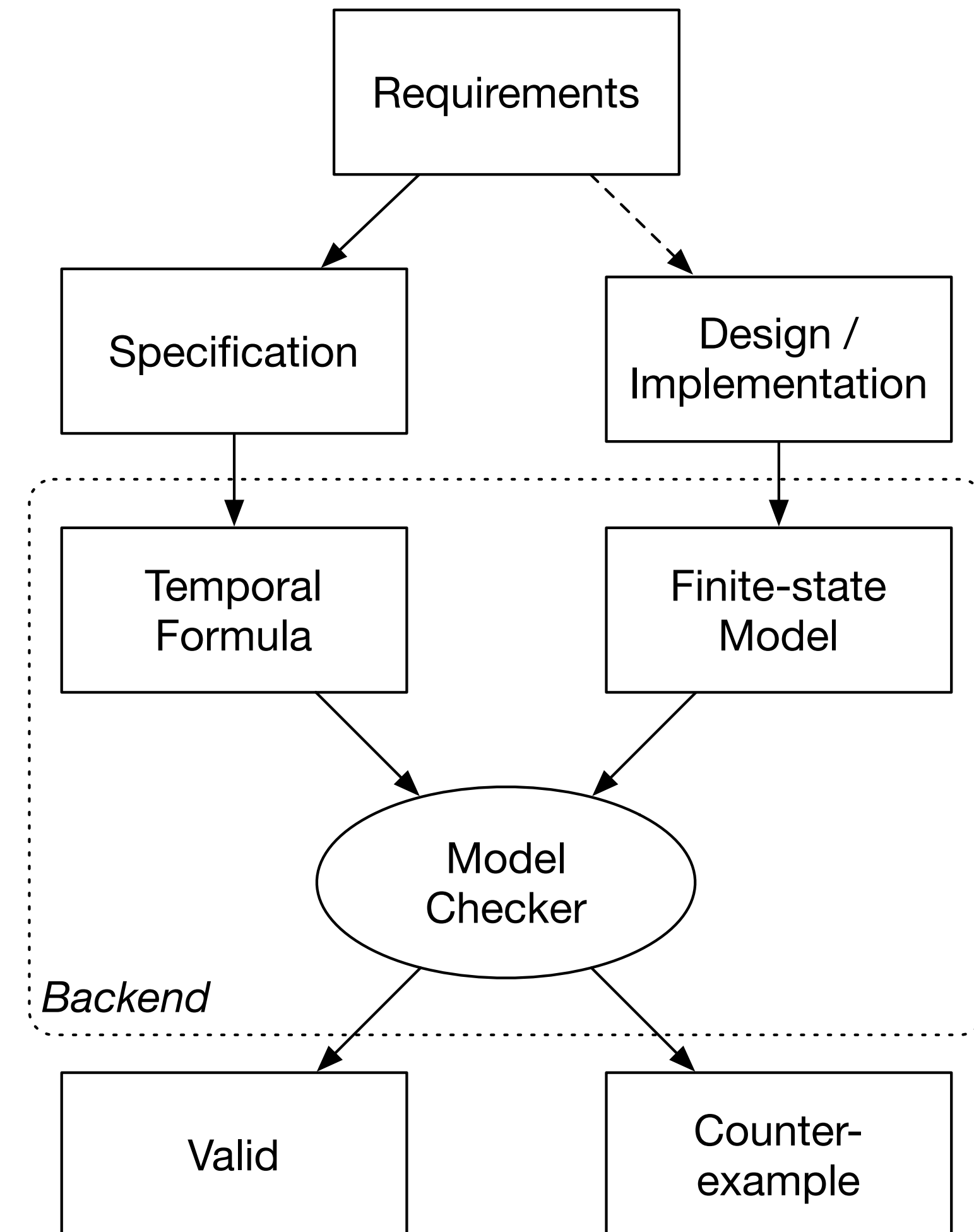


An Introduction to Model Checking

Nuno Macedo

Model Checking

- A classic method to check whether a property holds in a reactive system
- System as a **finite-state machine**: all states exhaustively explored
- Provided in a higher-level description (design model, implementation, ...)
- Requires a **bound** on the universe of discourse
- Properties in some kind of **propositional temporal logic**
- More expressive logics unfolded to propositional version within universe
- **Counter-examples** for invalid properties



System Model

- A model M is a transition system formalised as a Kripke structure (S, I, R, L) , over a set of atomic propositions P :
 - S is a finite set of states
 - $I \subseteq S$ is the set of initial states
 - $R \subseteq S \times S$ is a left-total transition relation
 - $L: S \rightarrow 2^P$ is a function assigning atomic propositions to each state

System Paths

- A path π is an infinite sequence of states $s_0s_1s_2\dots$ where $\forall i \geq 0 : (s_i, s_{i+1}) \in R$
- For $i \geq 0$, π_i is the i -th state of the path, π^i is the suffix starting in i -th state
- We abuse notation and say $\pi \in M$ for any path in model M

Model example: Mutex

while true:

n1: // noncritical actions

w1: request semaphore

c1: // critical section

release semaphore

||

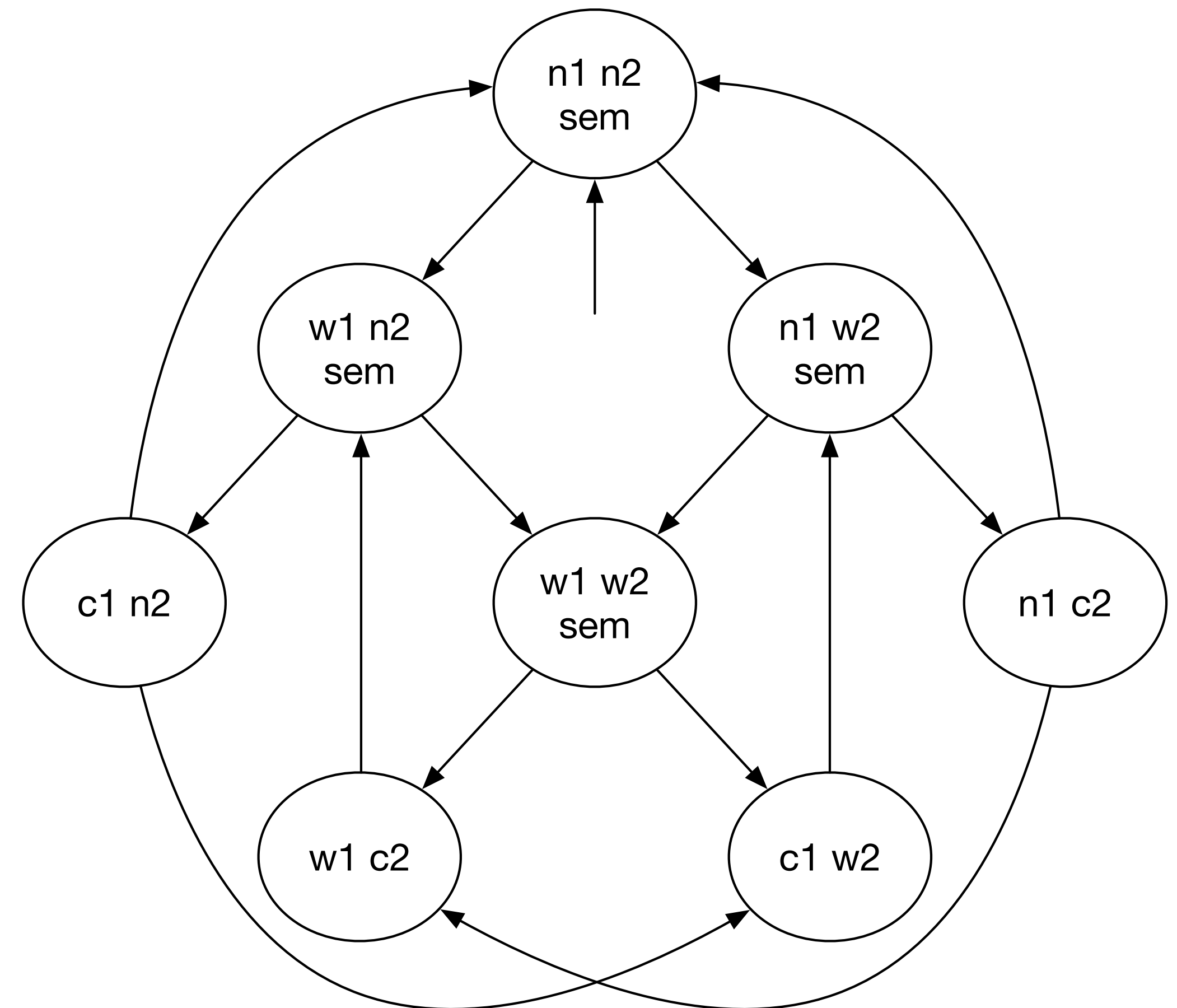
while true:

n2: // noncritical actions

w2: request semaphore

c2: // critical section

release semaphore

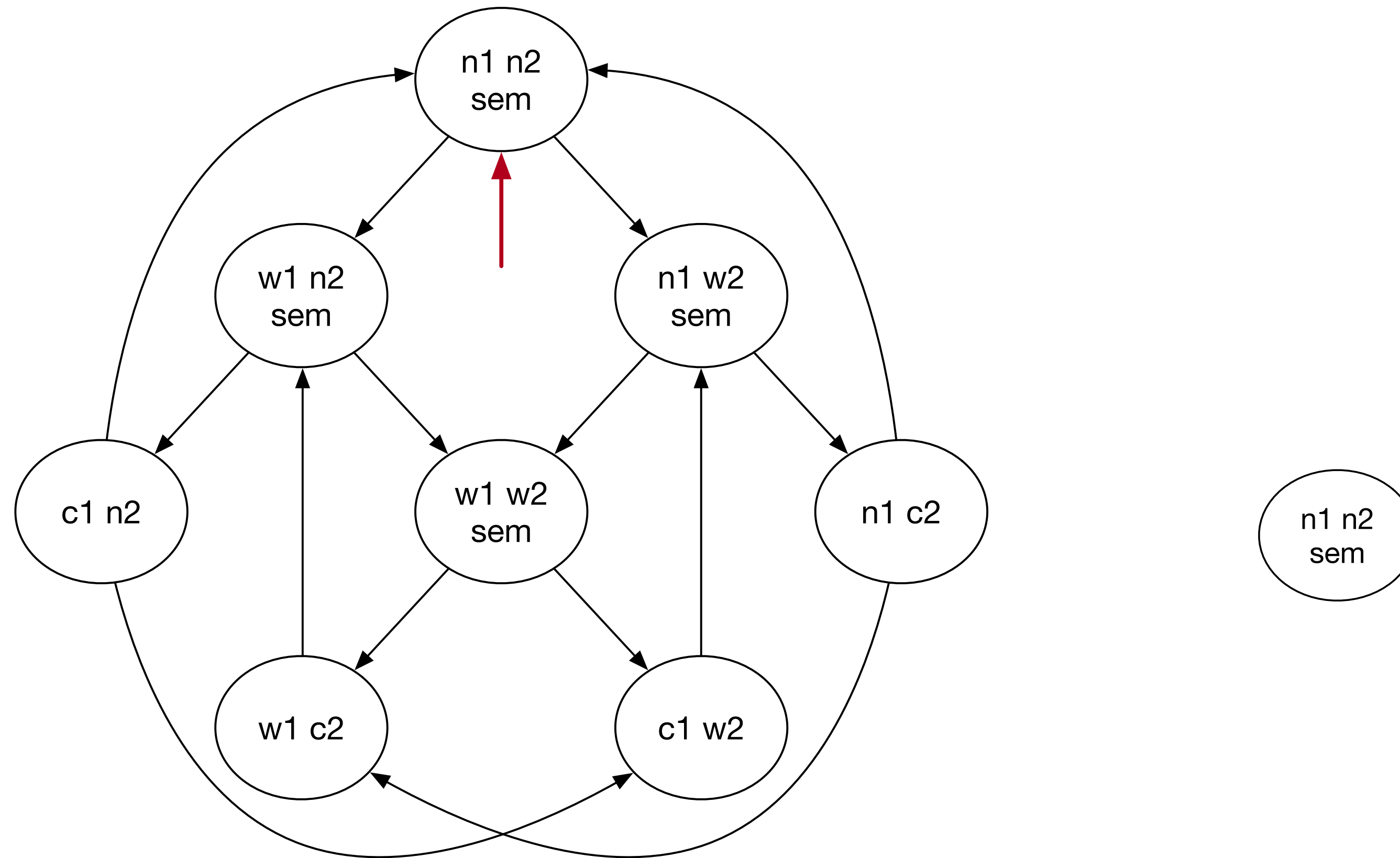


$P = \{n1, n2, w1, w2, c1, c2, sem\}$

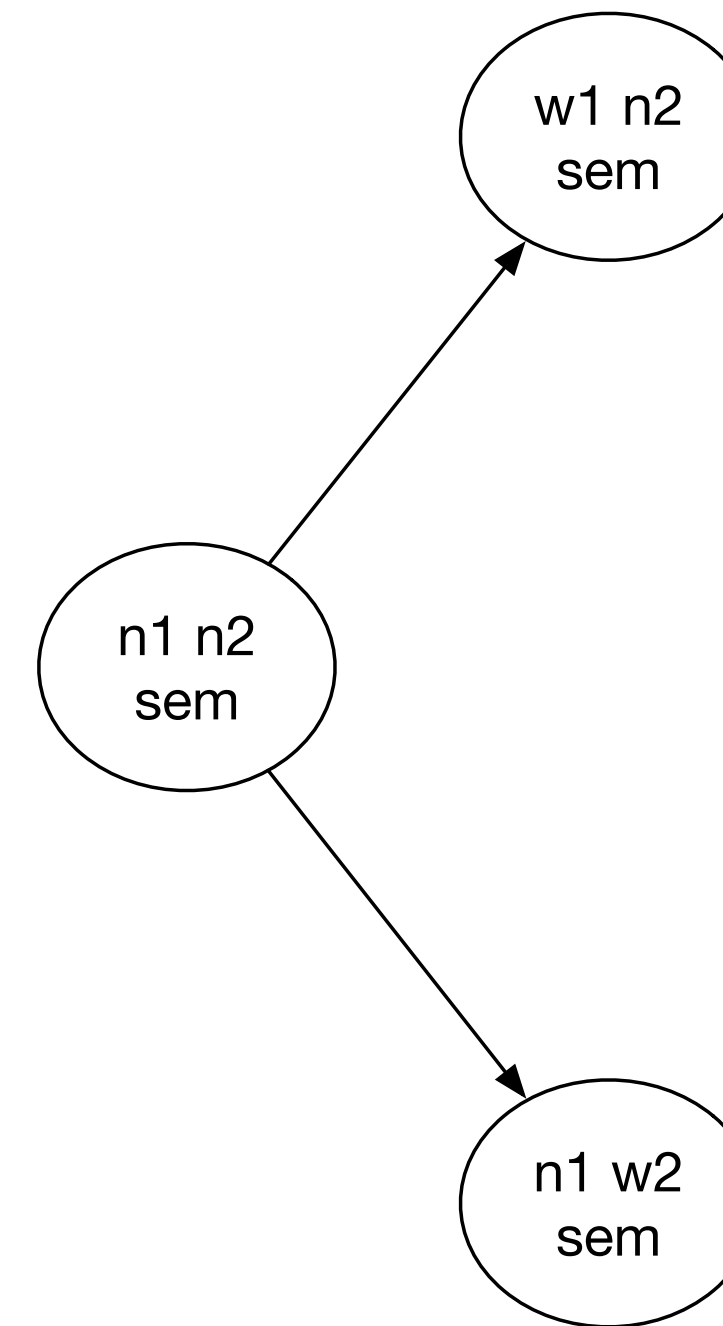
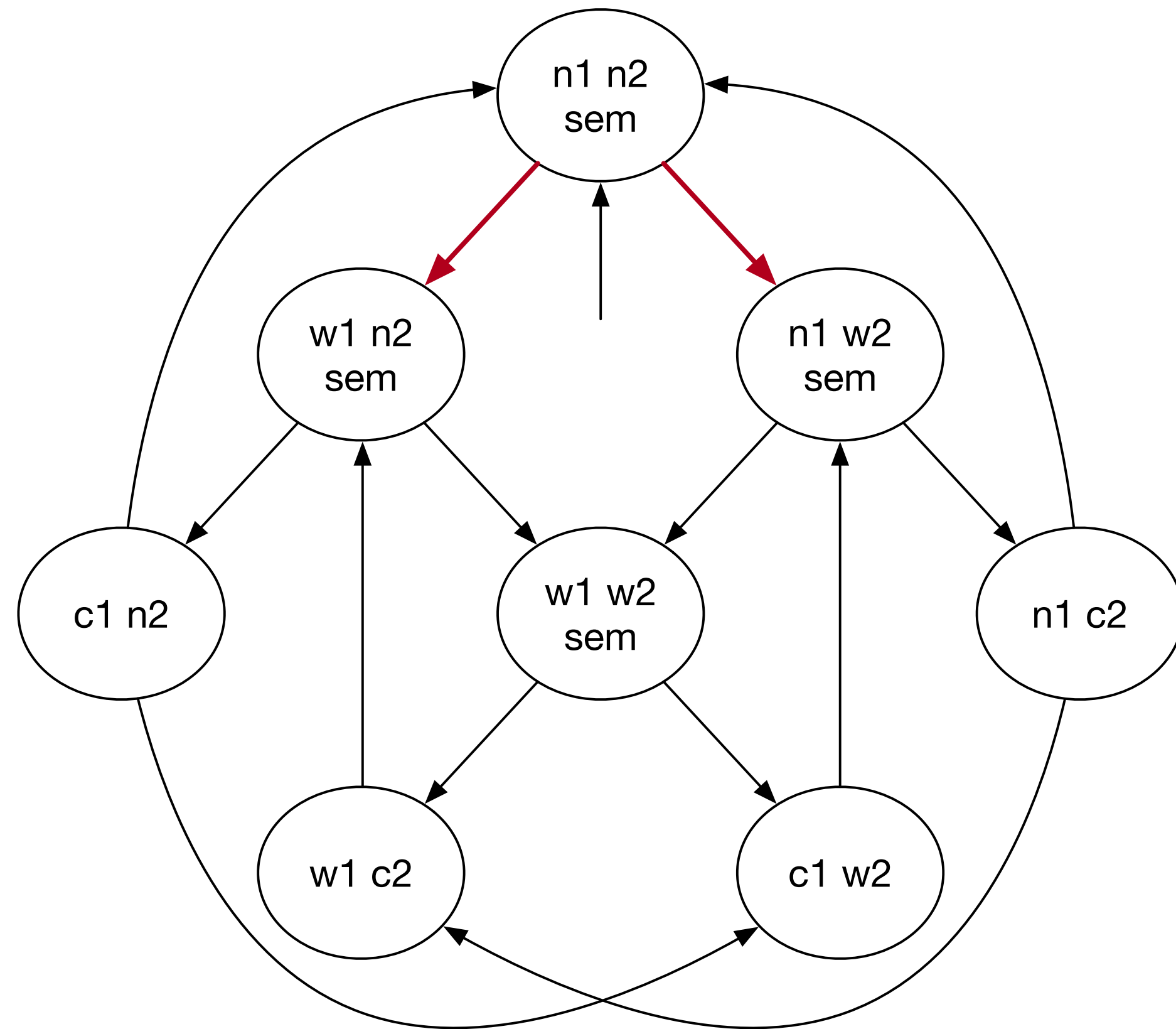
Temporal Logic

- Some properties can be checked by directly inspecting the Kripke structure (e.g., absence of deadlocks, or simple safety properties)
- To check more complex behaviours, relating multiple states, we need general-purpose **temporal logics**
- Standard temporal logics are **state-oriented** and **propositional**: they only consider the atomic propositions that are true in each state
- Two alternative models of time, supported by different logics:
 - **Linear Time**, the behaviour of a system is the set of all paths
 - **Branching Time**, the behaviour of the system is a set of computation trees

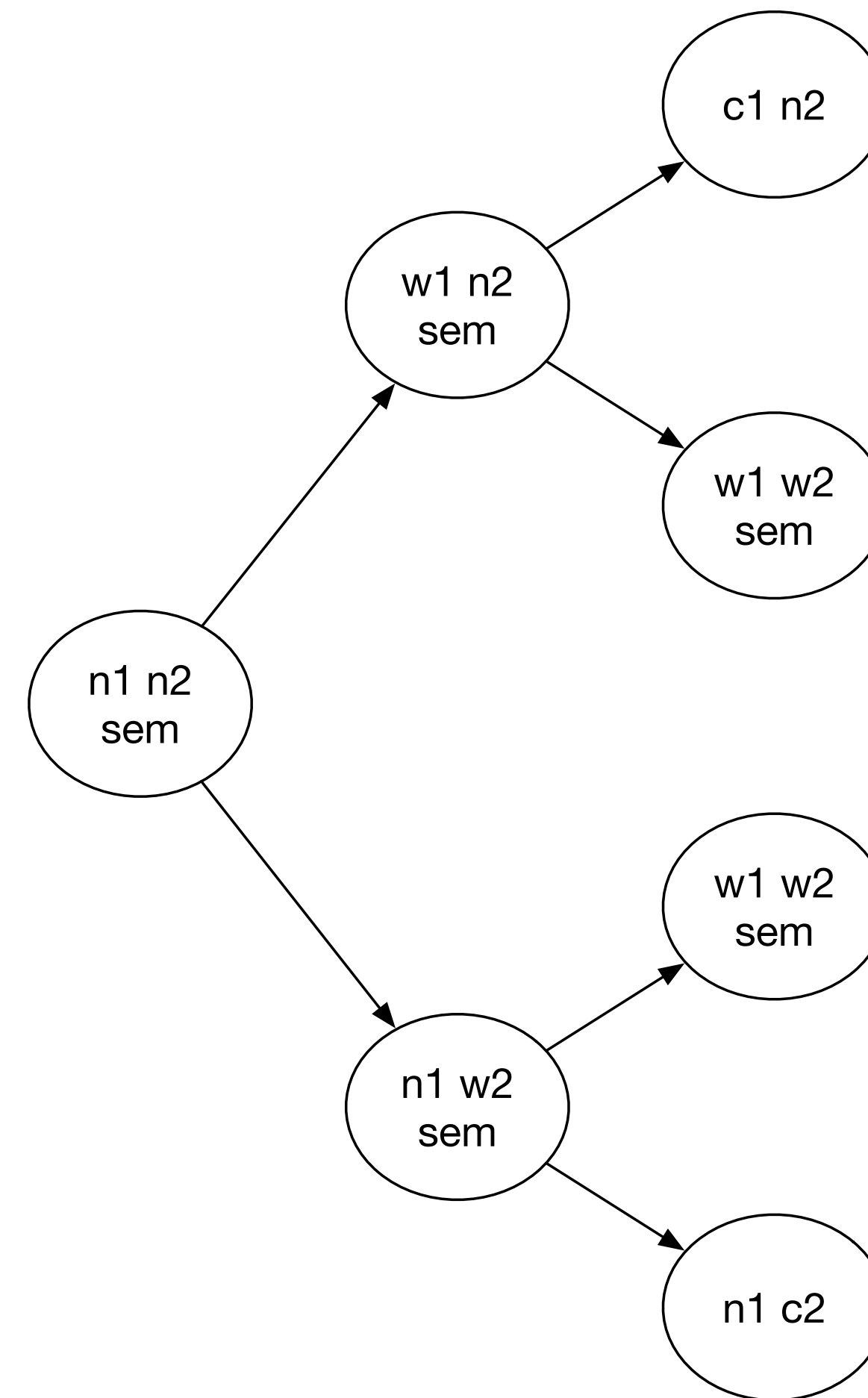
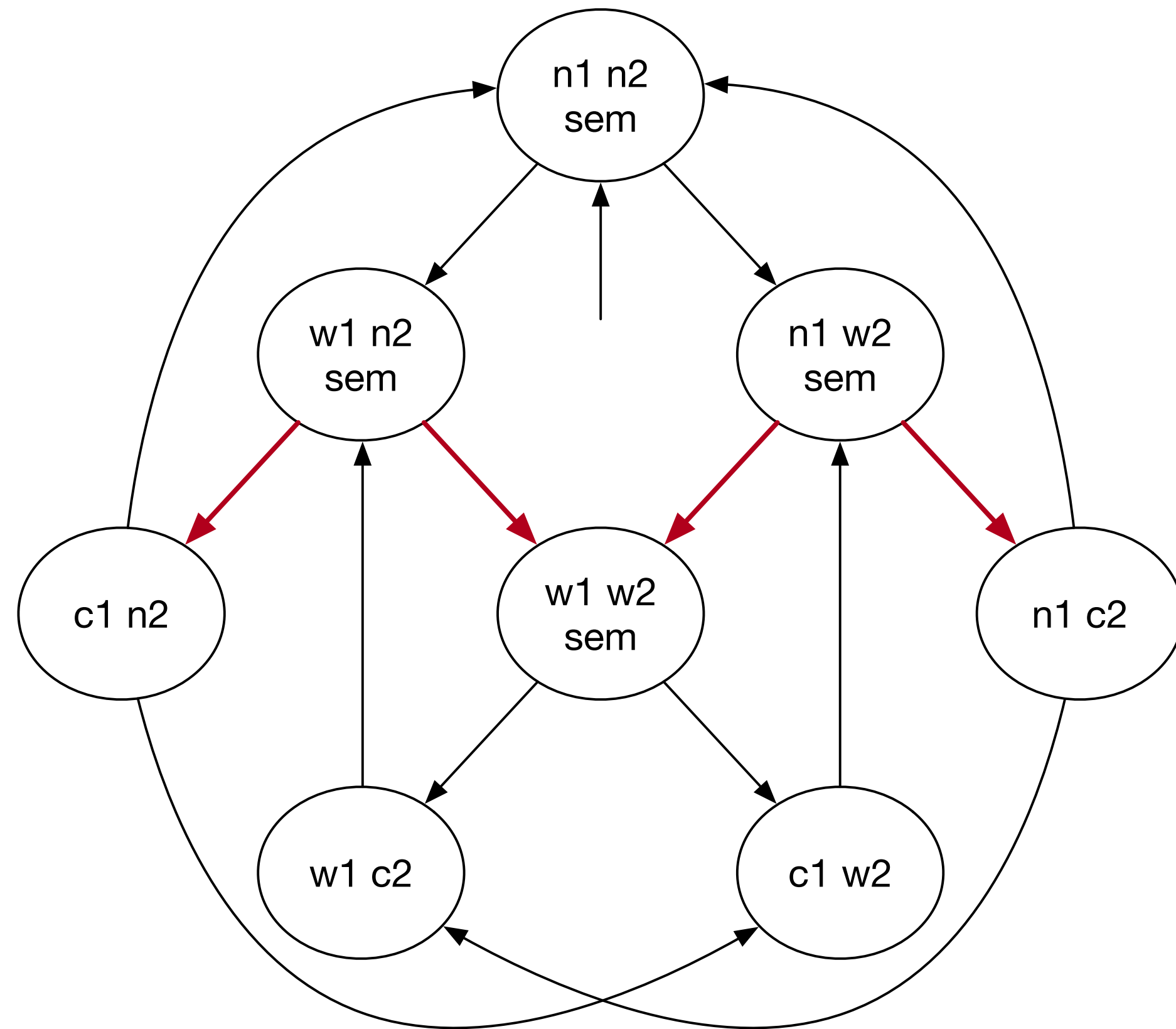
Branching Time



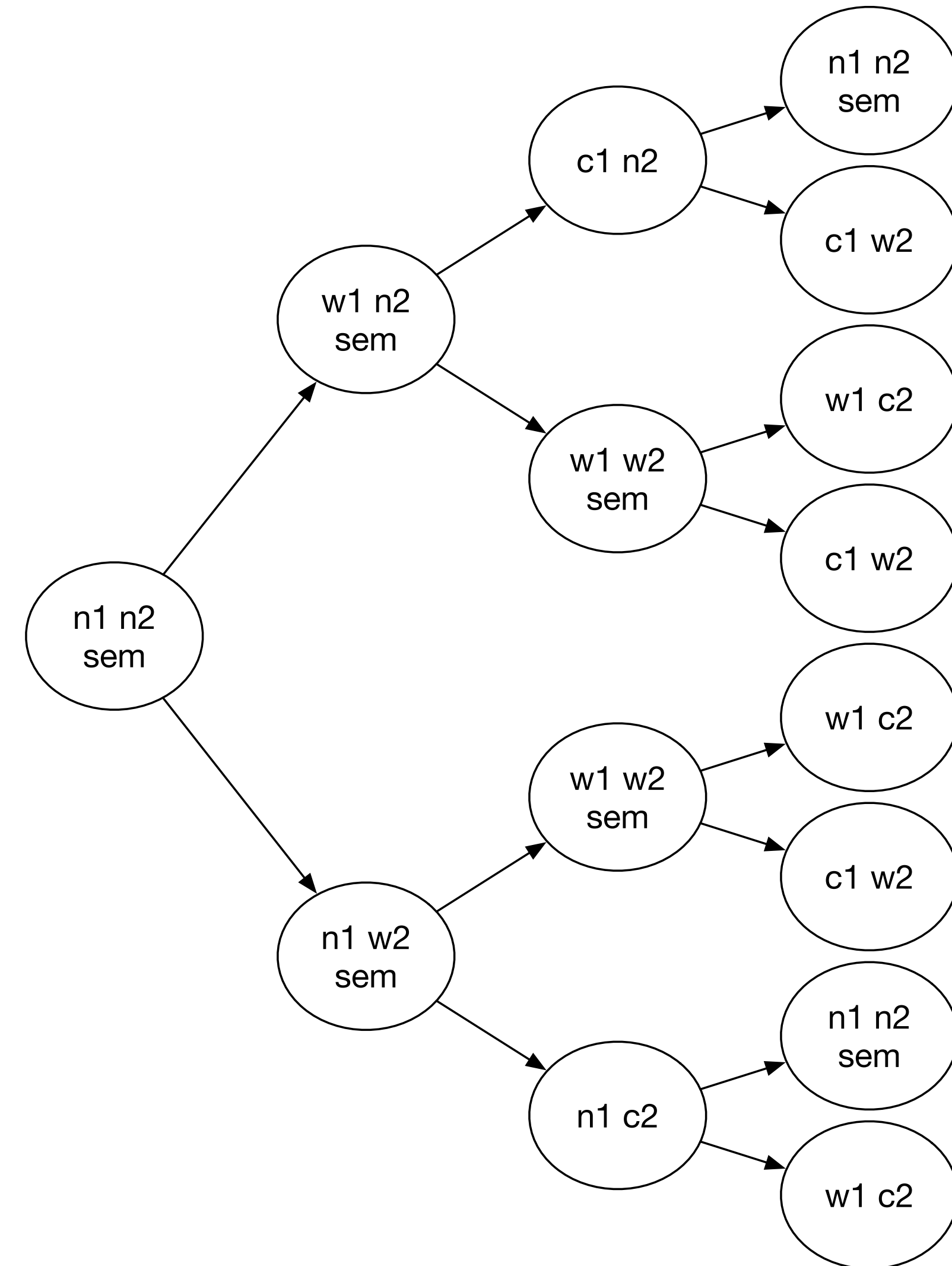
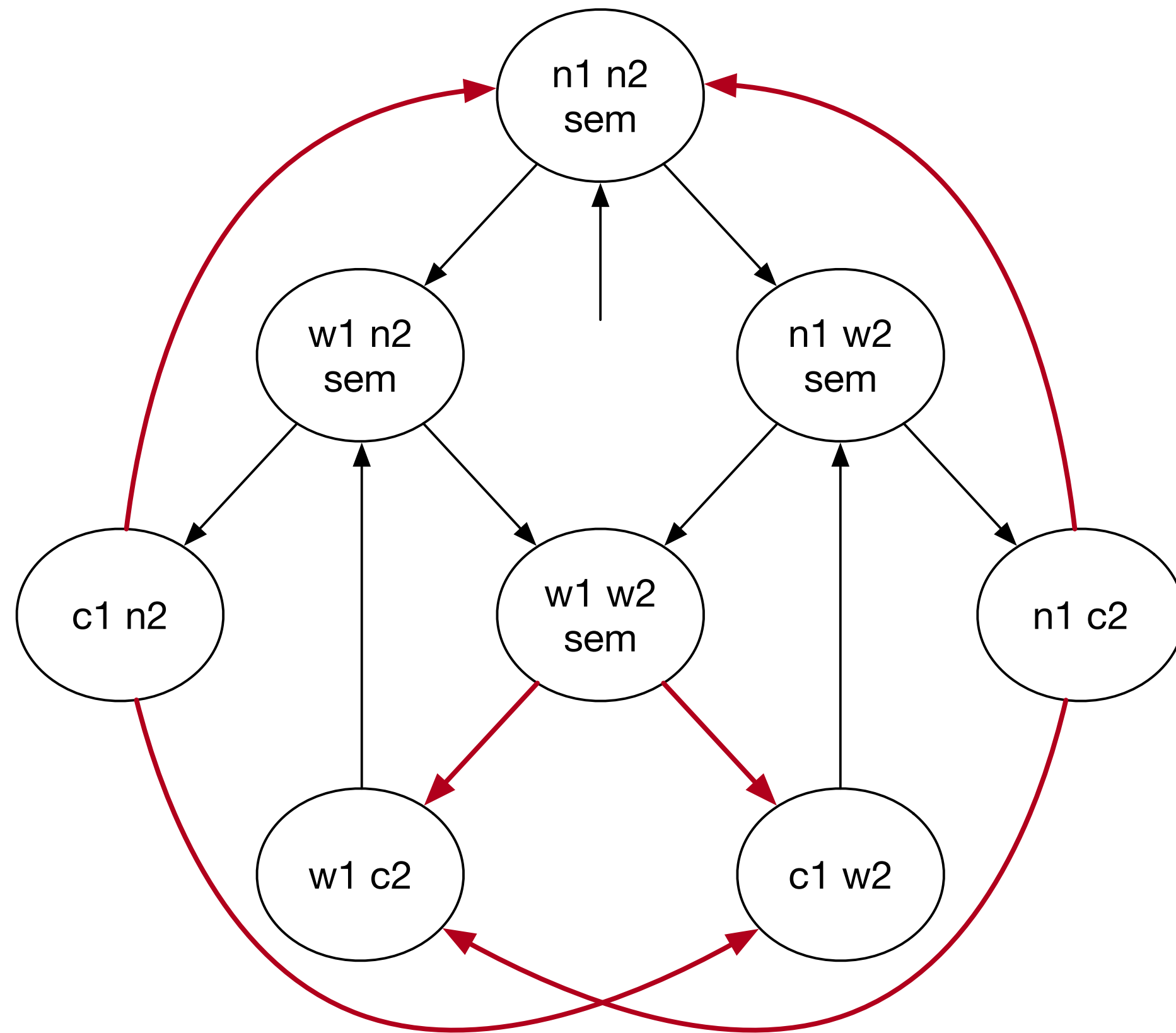
Branching Time



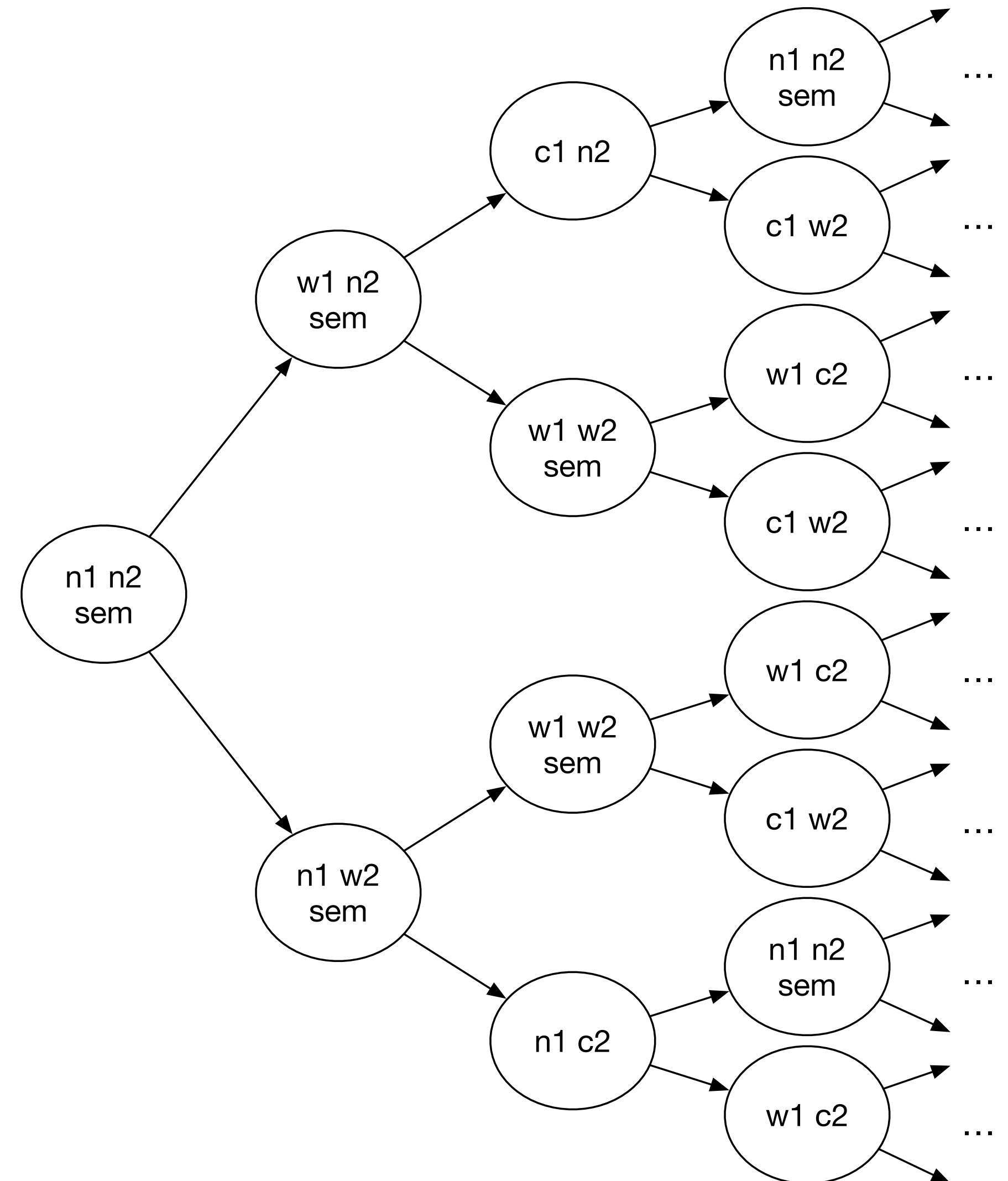
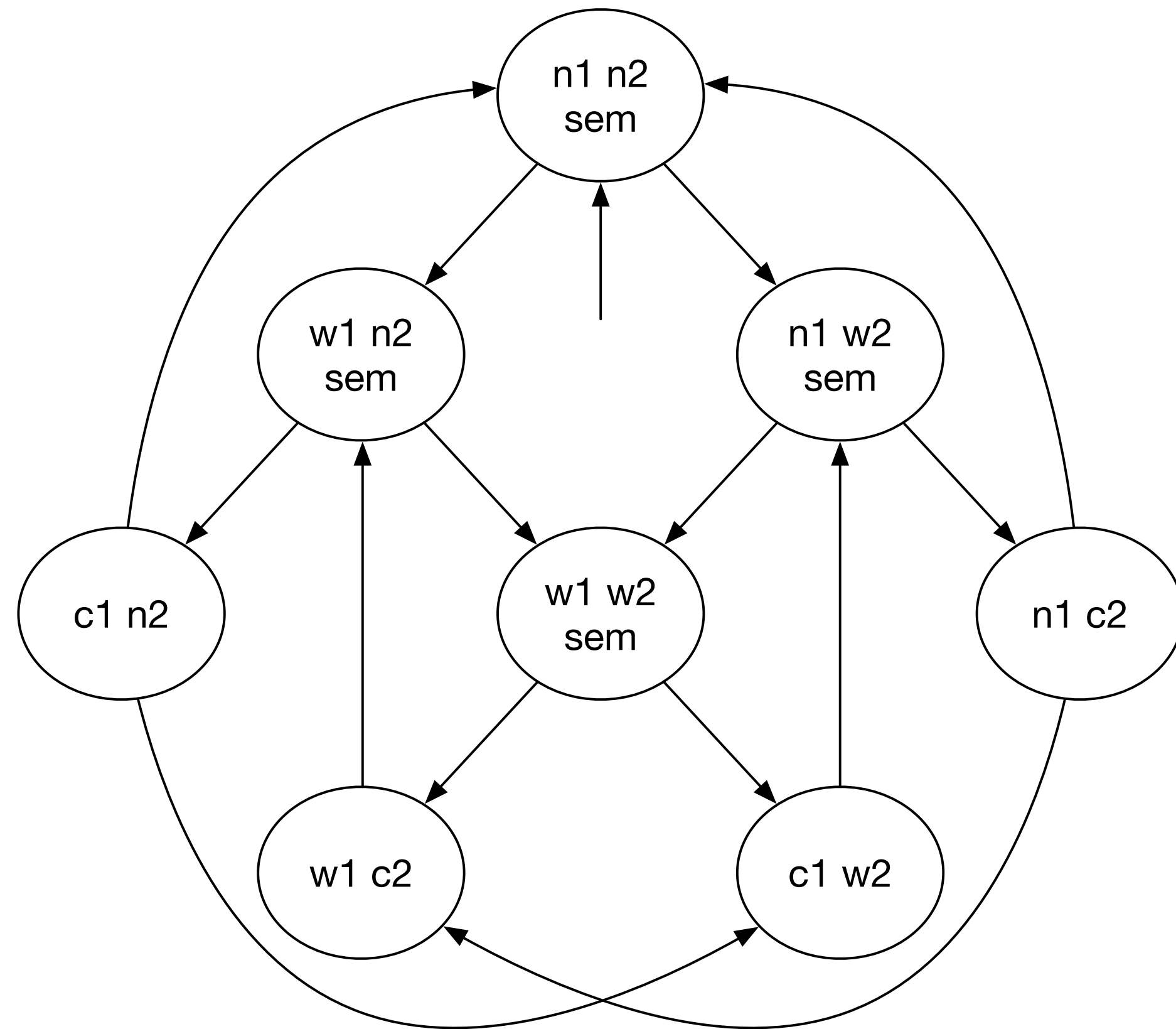
Branching Time



Branching Time



Branching Time



CTL

- **Computational Tree Logic** (CTL) is a branching time logic
- Besides the logical operators, CTL also has:
 - Path quantifiers
 - Temporal operators

CTL Syntax

- **X** (or \circ), **G** (or \square), **F** (or \diamond) and **U** are temporal operators:
 - **X** ϕ ϕ holds in the ne**X**t state
 - **G** ϕ ϕ always (or **G**lobally) holds
 - **F** ϕ ϕ eventually (or in the **F**uture) holds
 - ϕ **U** ψ ψ eventually holds and ϕ holds **U**ntil then
- **E** (or \exists) **A** (or \forall) are path quantifiers:
 - **A** ϕ ϕ holds in **all** computation paths starting in the current state
 - **E** ϕ ϕ holds in **some** computation path starting in the current state
- Path quantifiers must always be followed by a temporal operator (path formula)

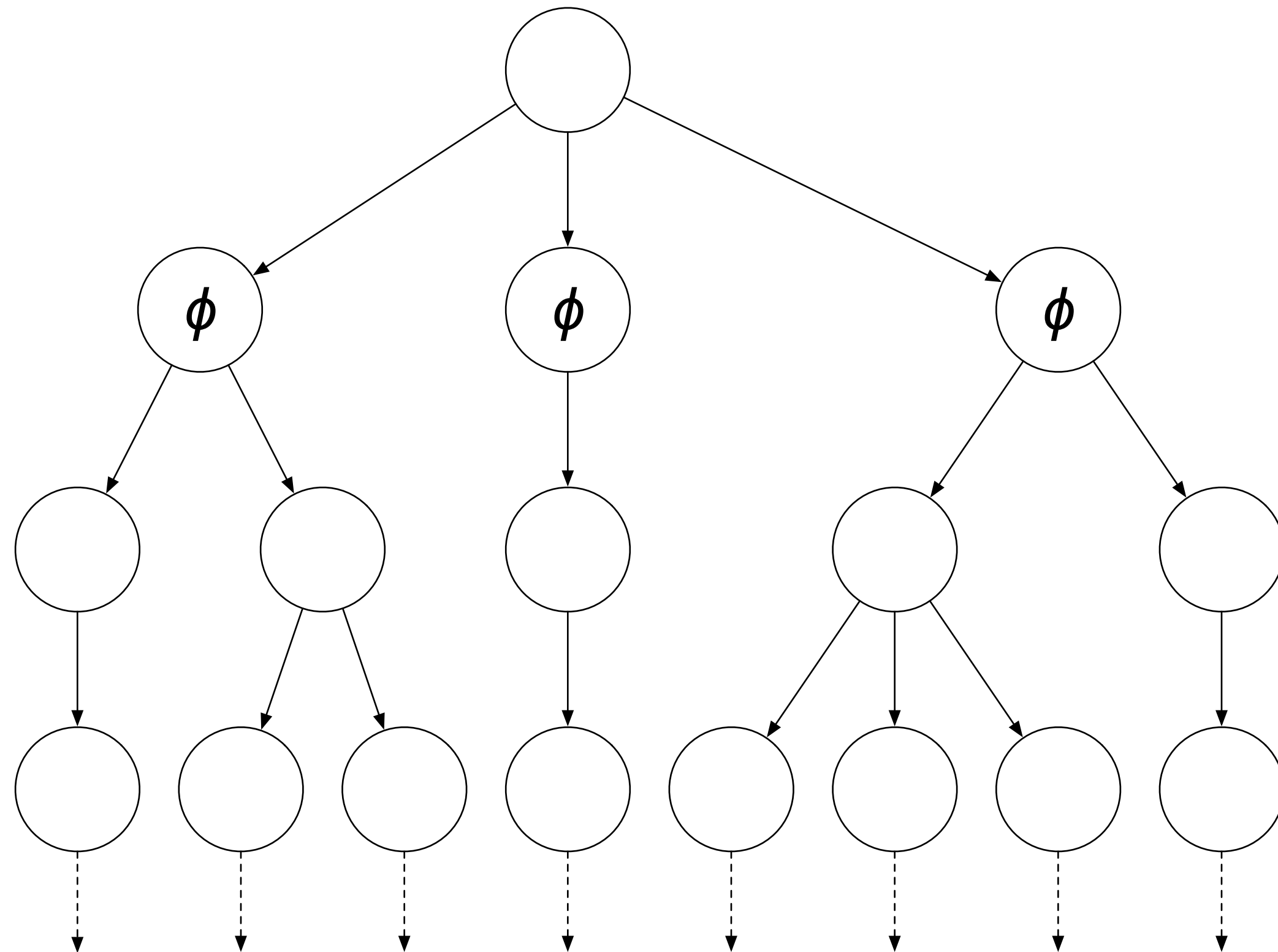
CTL Syntax

$$\begin{aligned} \phi, \psi ::= & \mathbf{AX} \phi \mid \mathbf{EX} \phi \\ & \mid \mathbf{AG} \phi \mid \mathbf{EG} \phi \\ & \mid \mathbf{AF} \phi \mid \mathbf{EF} \phi \\ & \mid \mathbf{A}[\phi \mathbf{U} \psi] \mid \mathbf{E}[\phi \mathbf{U} \psi] \\ & \mid p \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \neg \phi \mid \top \mid \perp \end{aligned}$$

with $p \in P$ atomic propositions

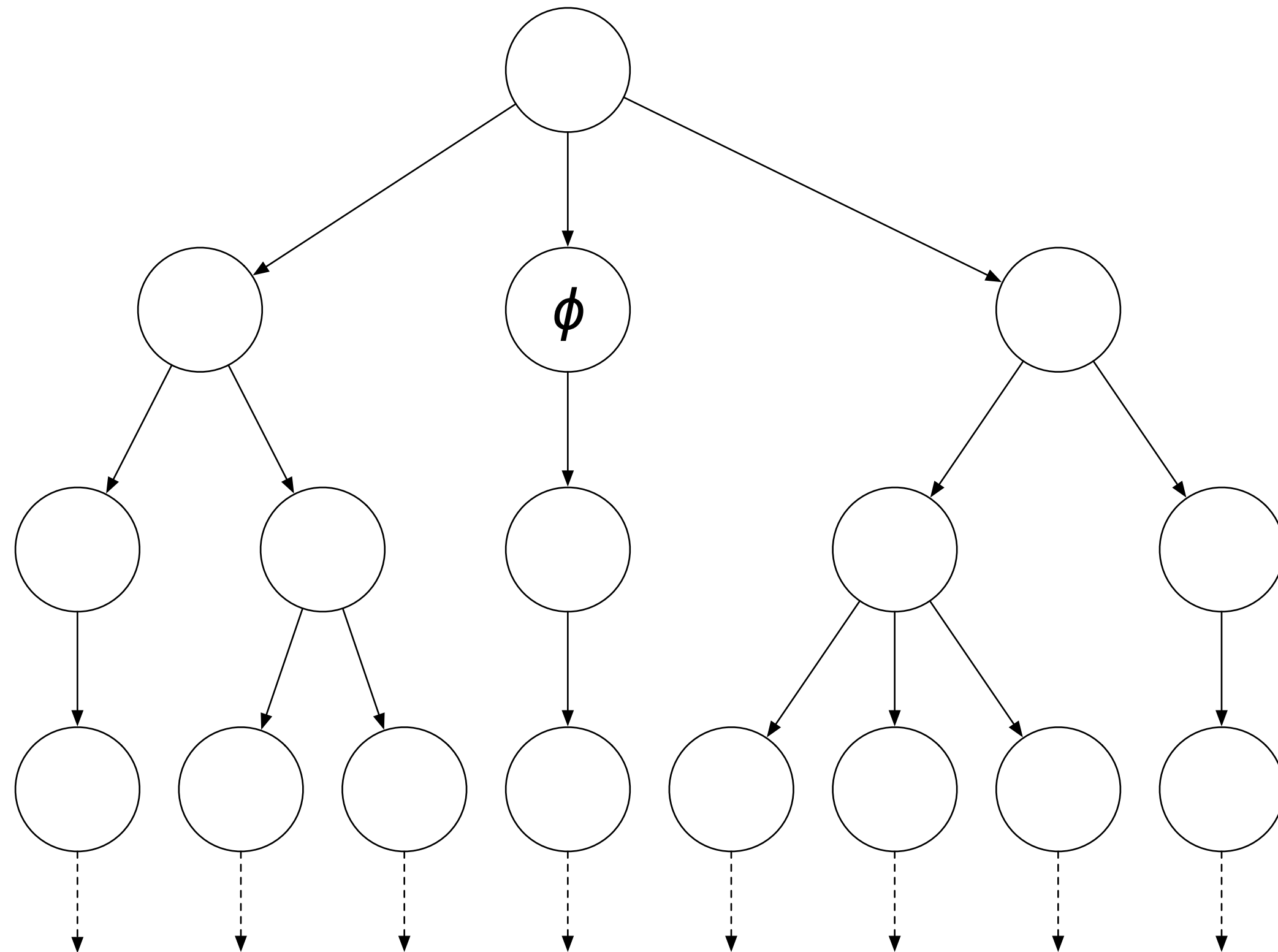
CTL Semantics

$AX \phi$



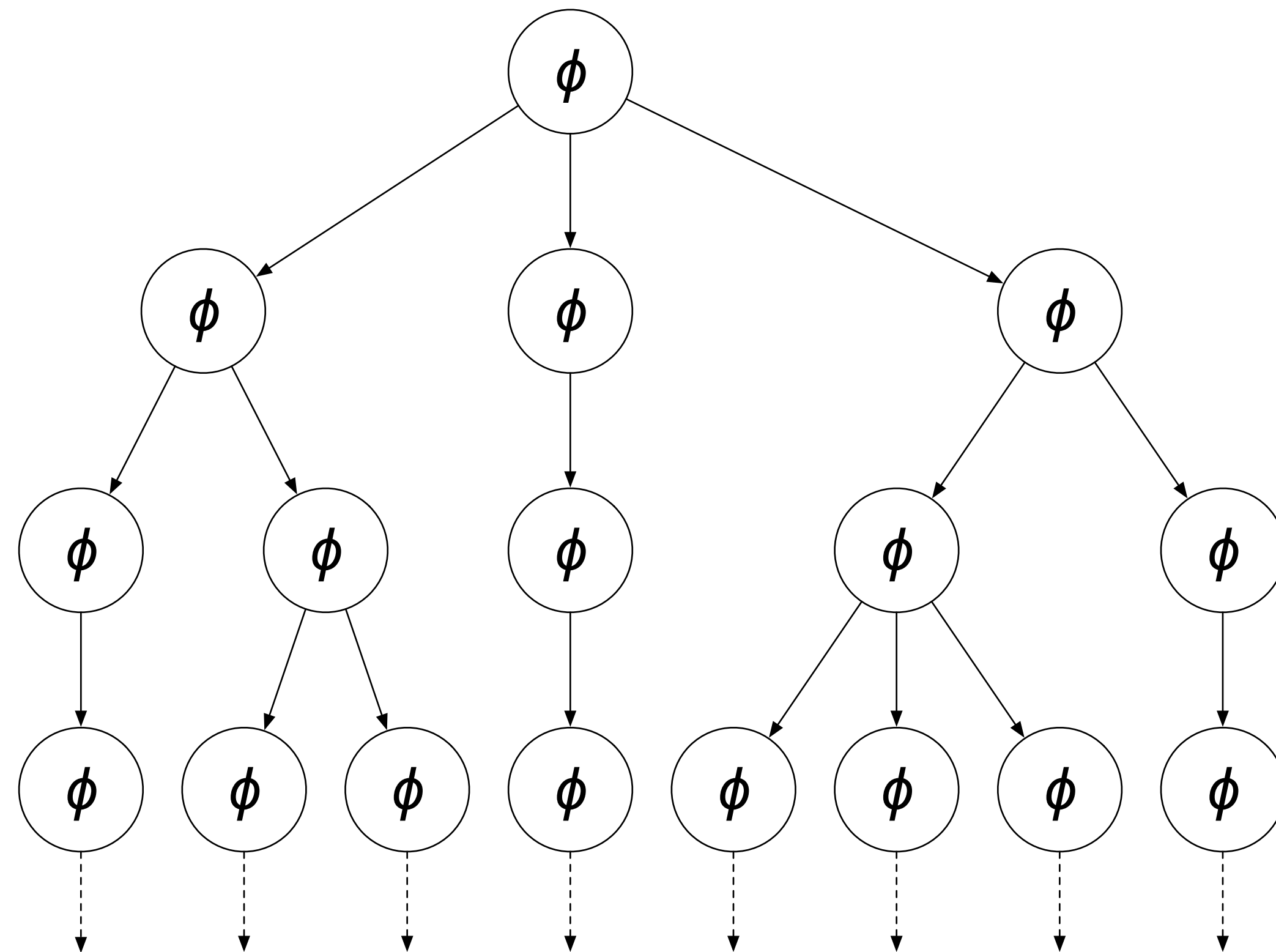
CTL Semantics

EX ϕ



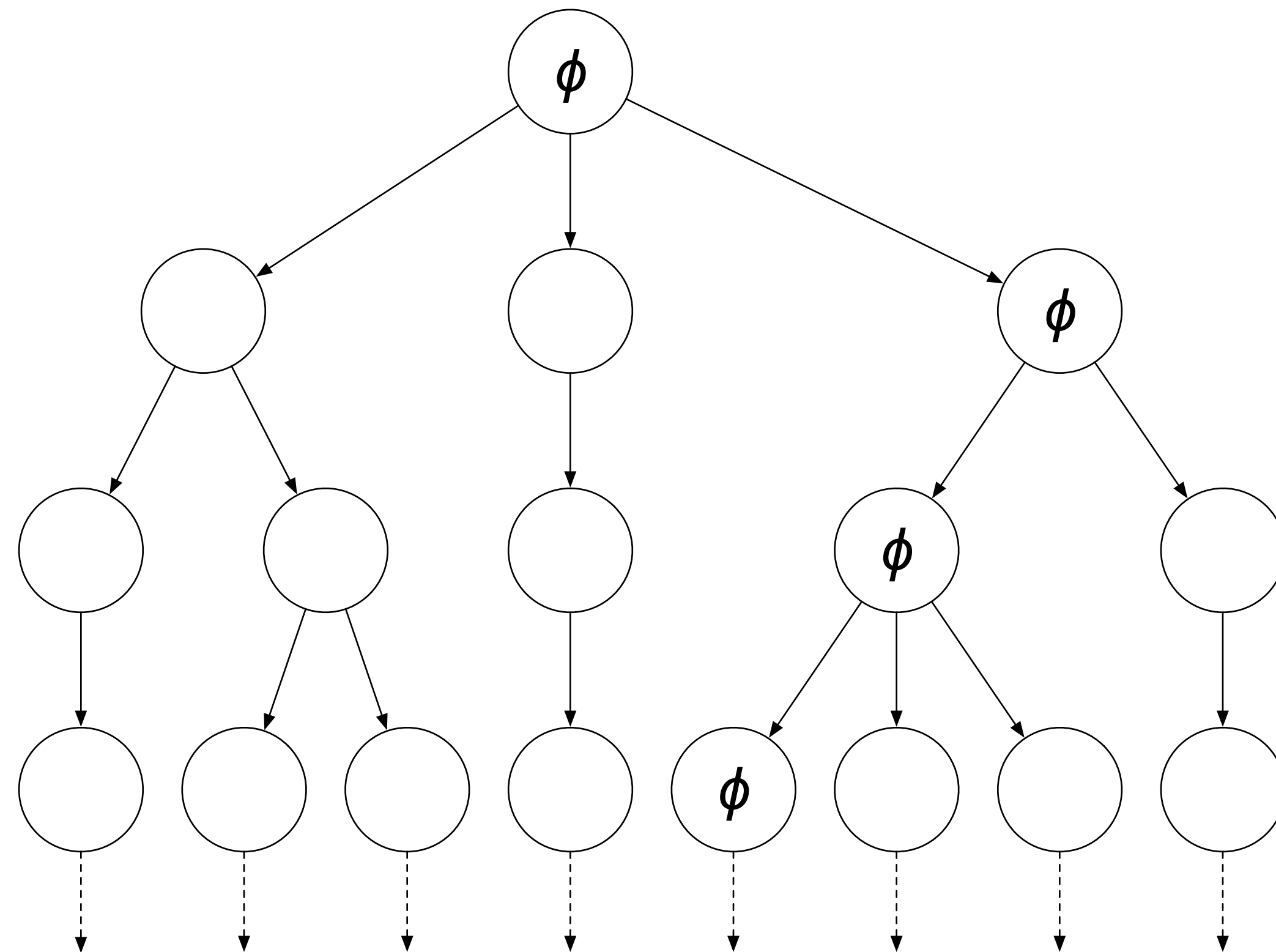
CTL Semantics

AG ϕ



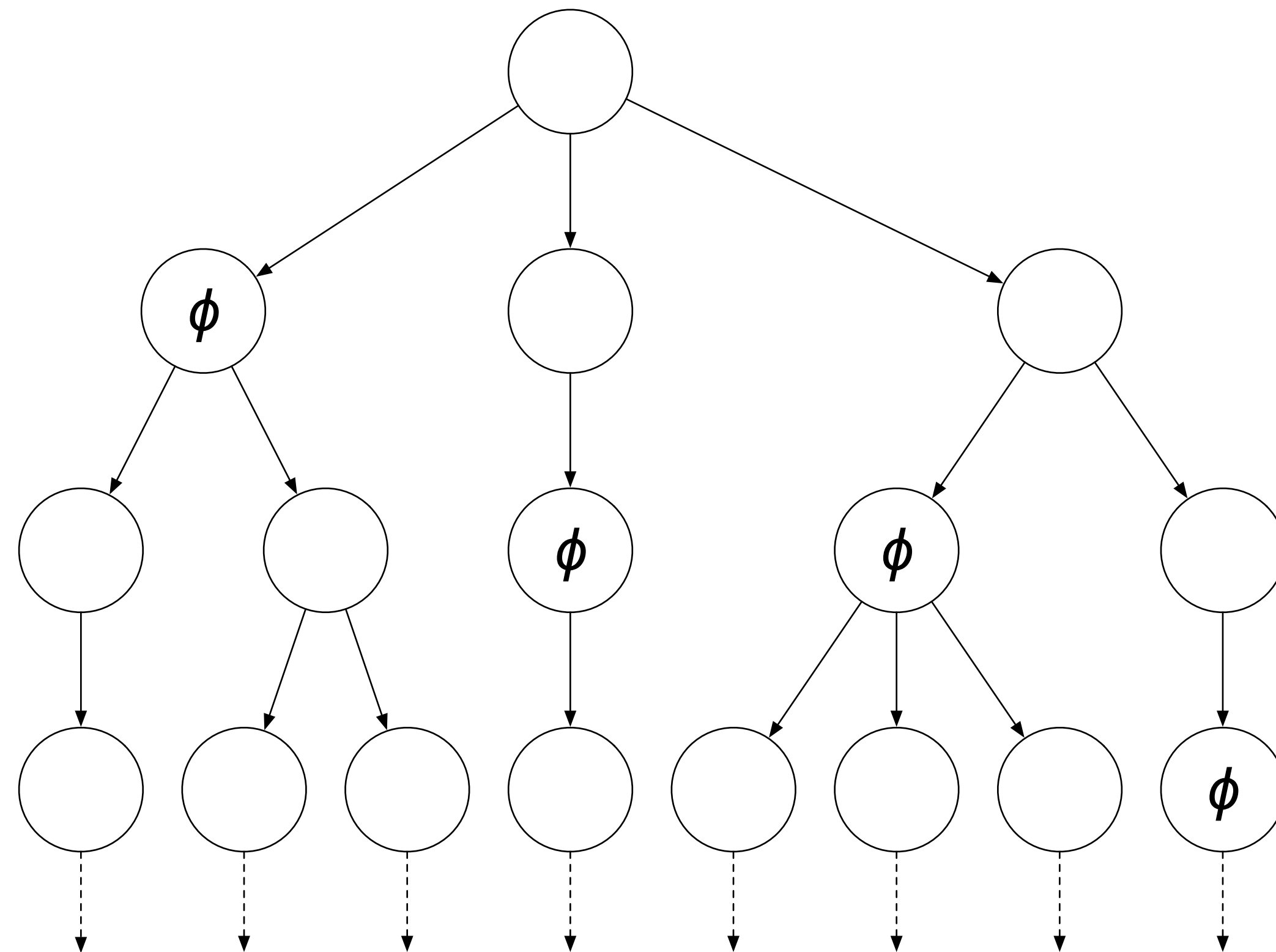
CTL Semantics

EG ϕ



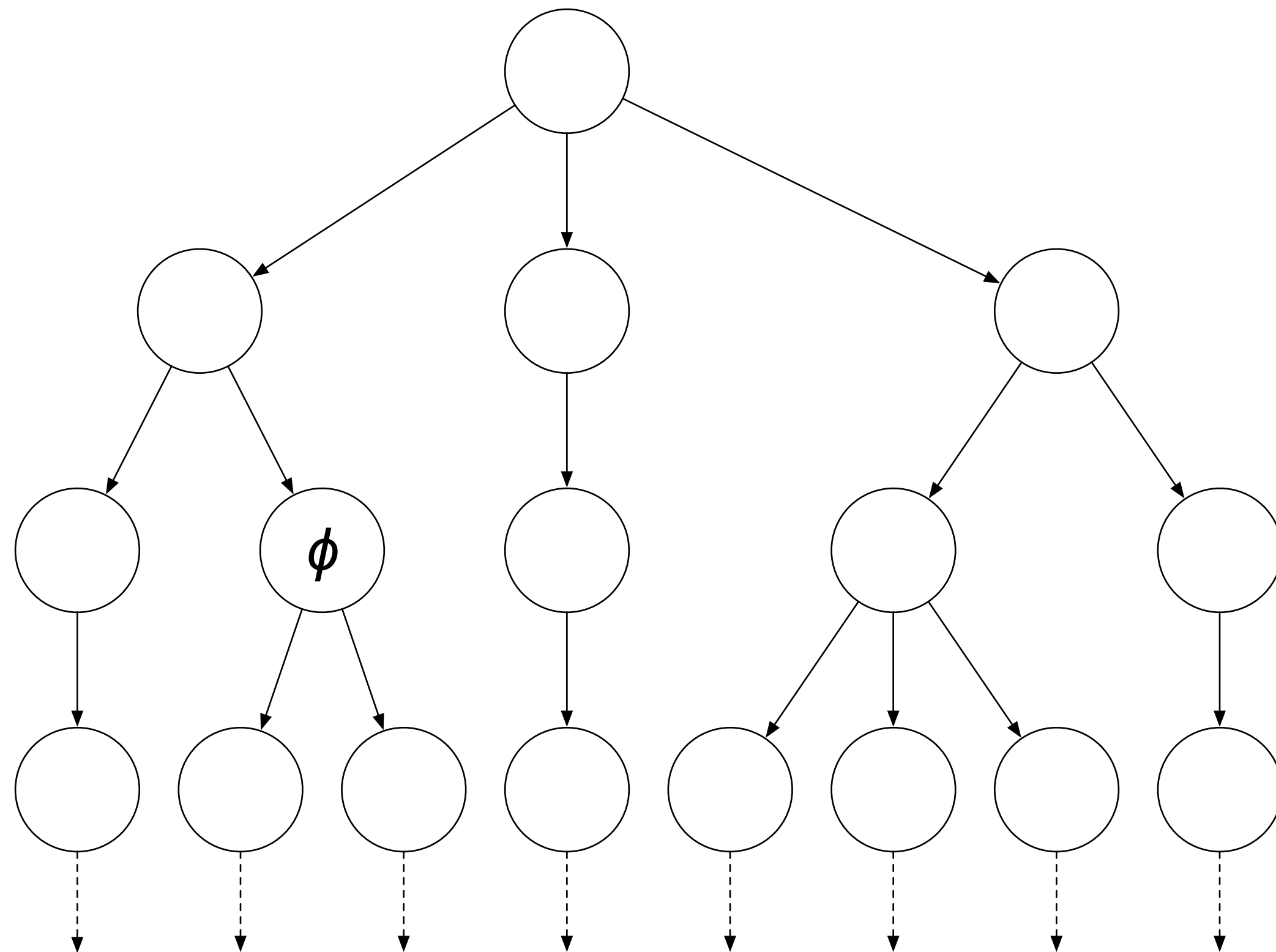
CTL Semantics

AF ϕ



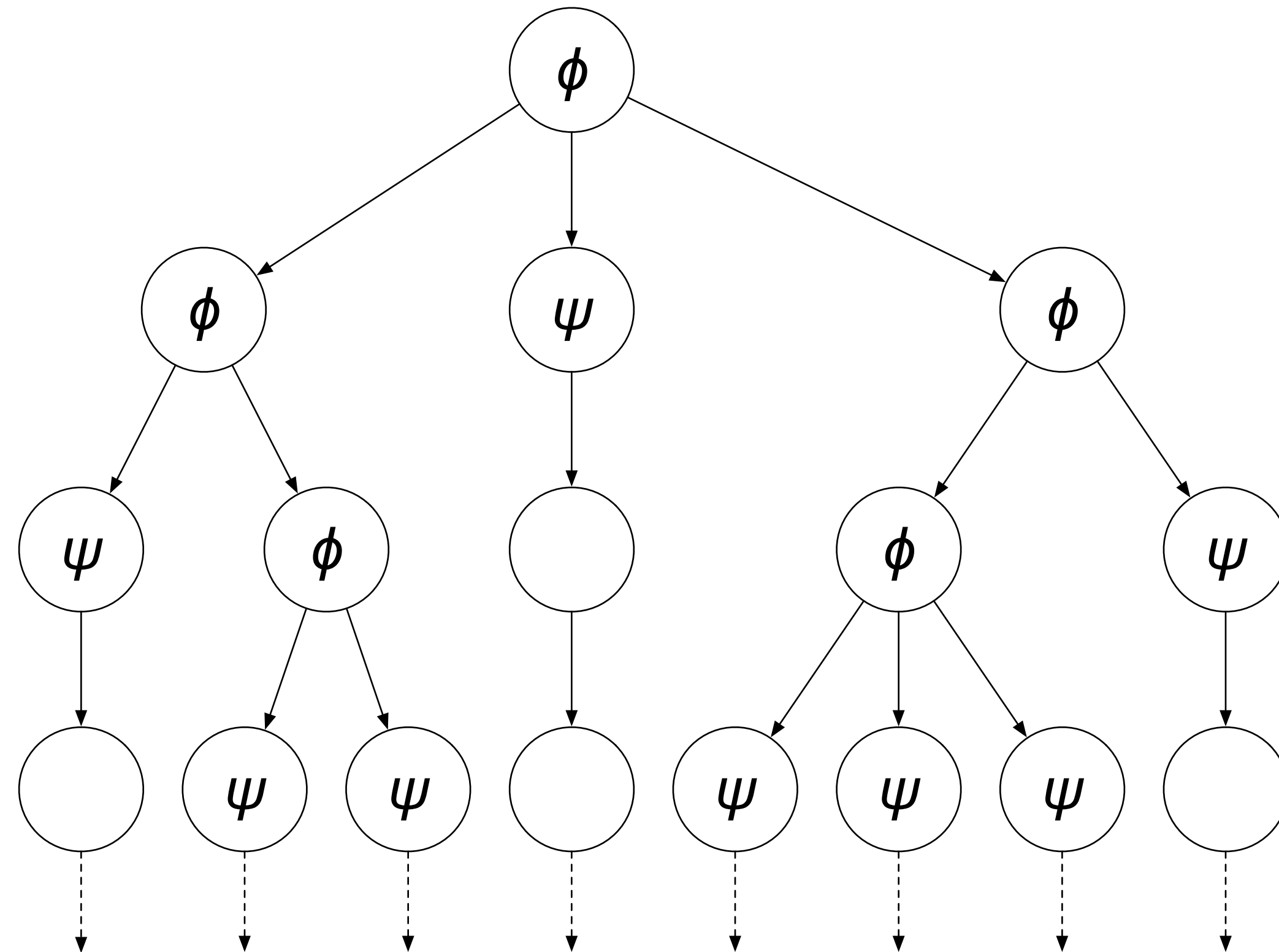
CTL Semantics

EF ϕ



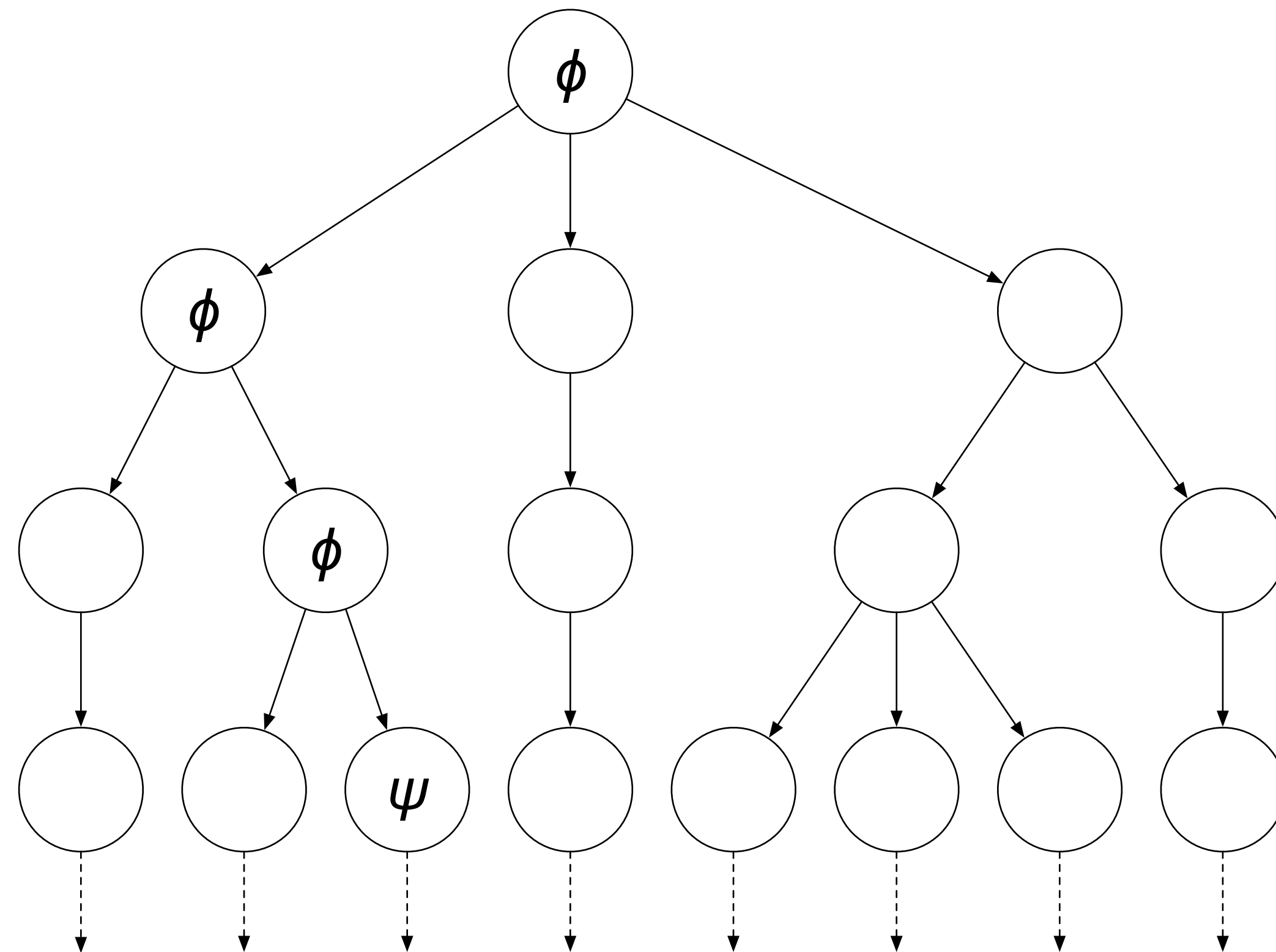
CTL Semantics

$A[\phi U \psi]$



CTL Semantics

$E[\phi U \psi]$



CTL Semantics

- If a CTL formula ϕ holds for a Kripke structure $M = (S, I, R, L)$ we say

$$M \models \phi$$

- $M \models \phi$ iff for all initial states $s \in I$ we have $M, s \models \phi$

- Minimal CTL subset: \top , \vee , \neg , **EG**, **EU**, **EX**

- $\mathbf{EF}\phi \quad \equiv \quad \mathbf{E}[\top \mathbf{U}\phi]$

- $\mathbf{AX}\phi \quad \equiv \quad \neg \mathbf{EX}(\neg\phi)$

- $\mathbf{AG}\phi \quad \equiv \quad \neg \mathbf{EF}(\neg\phi)$

- $\mathbf{AF}\phi \quad \equiv \quad \neg \mathbf{EG}(\neg\phi)$

- $\mathbf{A}[\phi \mathbf{U}\psi] \quad \equiv \quad \neg(\mathbf{E}[(\neg\psi) \mathbf{U}\neg(\phi \vee \psi)] \vee \mathbf{EG}(\neg\psi))$

CTL Semantics

$$M, s \models p \quad \equiv \quad p \in L(s)$$

$$M, s \models \top \quad \equiv \quad \top$$

$$M, s \models \neg\phi \quad \equiv \quad M, s \not\models \phi$$

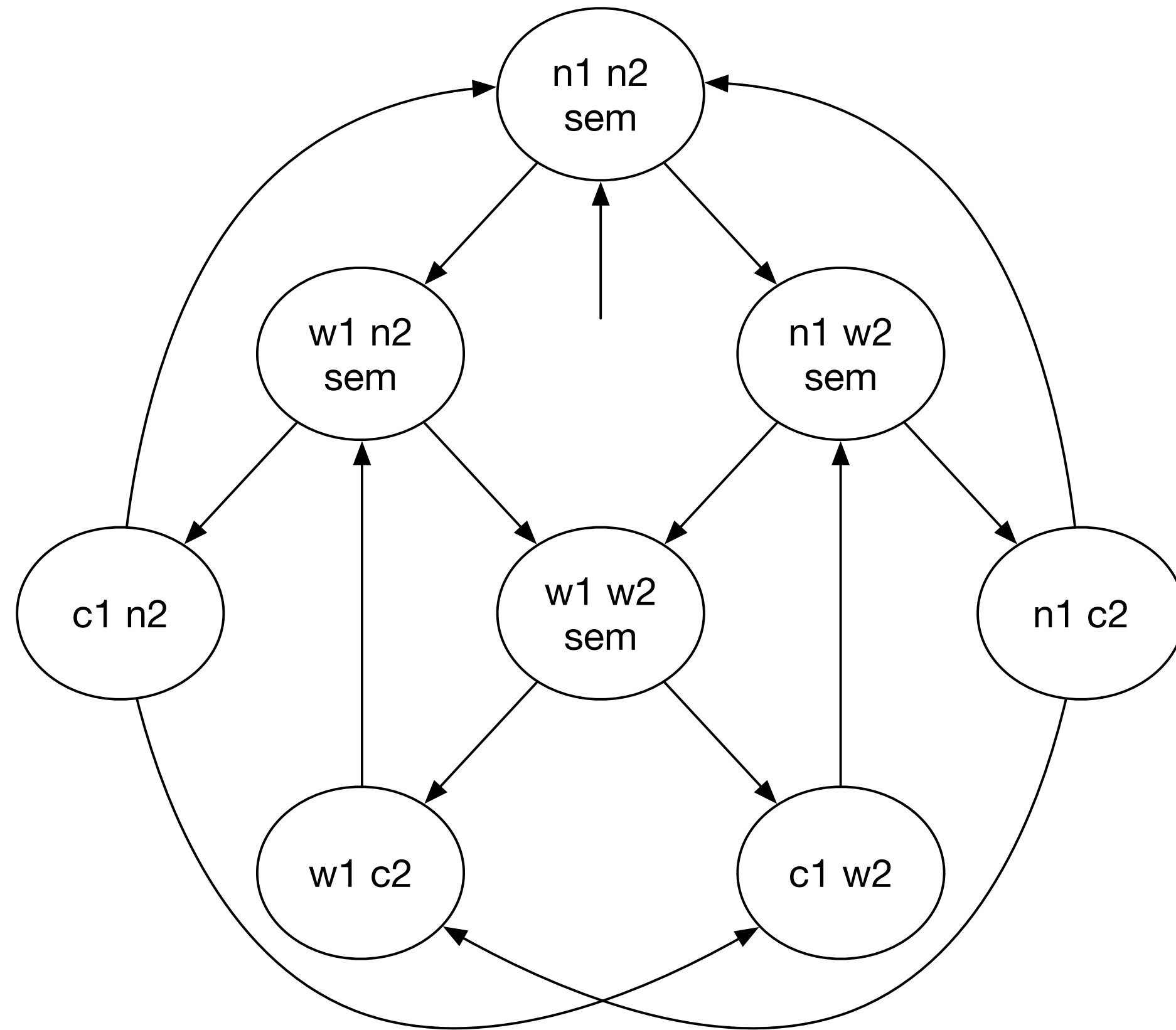
$$M, s \models \phi \vee \psi \quad \equiv \quad M, s \models \phi \text{ or } M, s \models \psi$$

$$M, s \models \mathbf{EX}\phi \quad \equiv \quad \exists \pi \in M . \pi_0 = s \text{ and } M, \pi_1 \models \phi$$

$$M, s \models \mathbf{EG}\phi \quad \equiv \quad \exists \pi \in M . \pi_0 = s \text{ and } \forall i \geq 0 . M, \pi_i \models \phi$$

$$M, s \models \mathbf{E}[\phi \mathbf{U} \psi] \quad \equiv \quad \exists \pi \in M . \pi_0 = s \text{ and } \exists i \geq 0 . (M, \pi_i \models \psi \text{ and } \forall 0 \leq j < i \text{ } M . \pi_j \models \phi)$$

CTL Examples

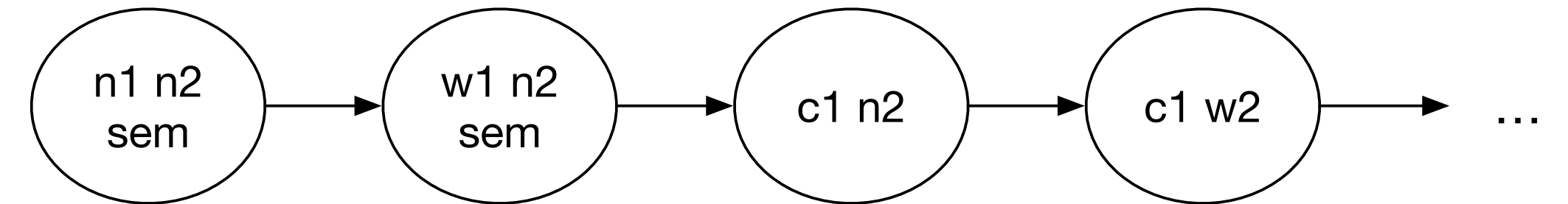
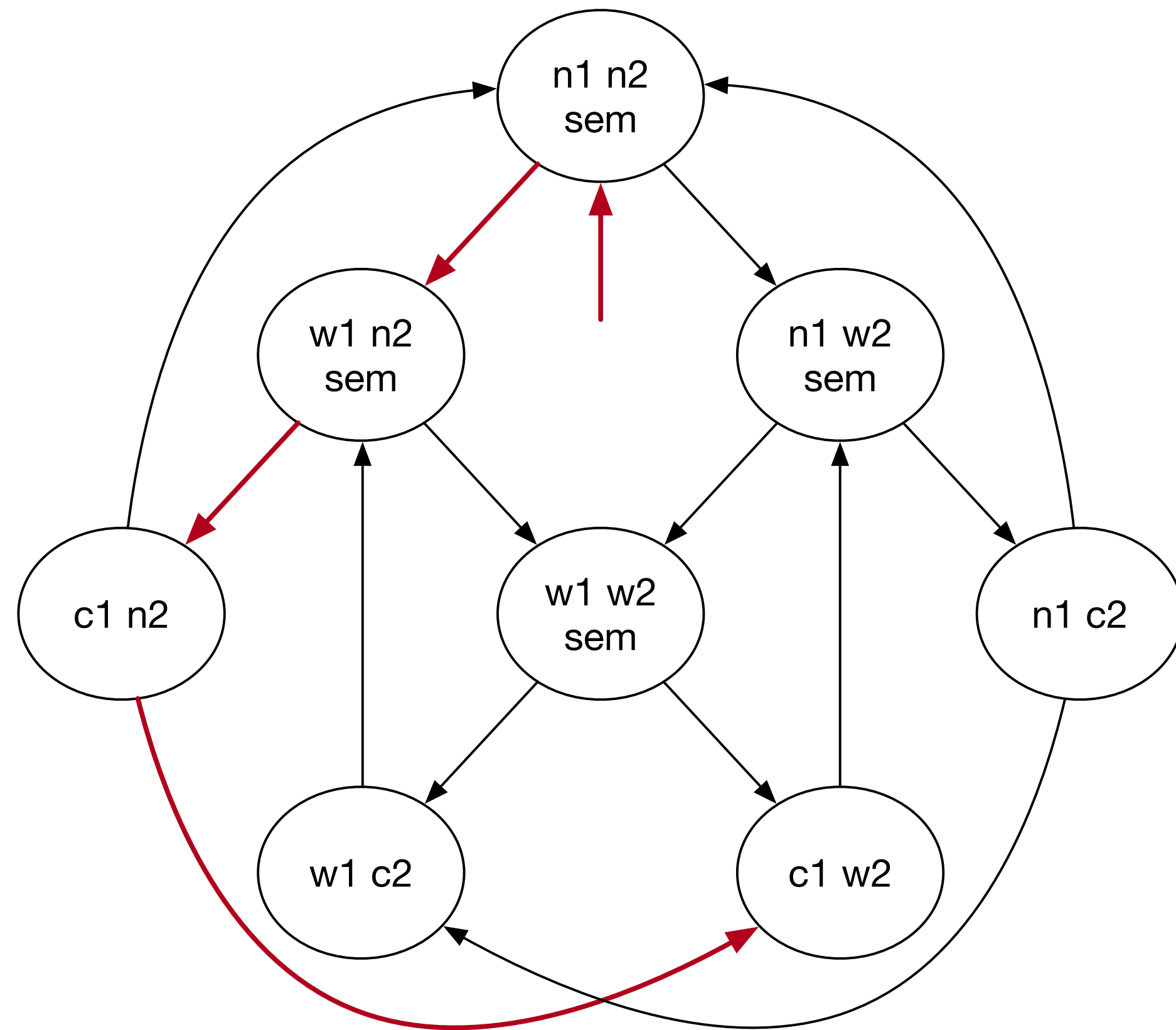


Mutual exclusion: $\text{AG } \neg(c1 \wedge c2)$

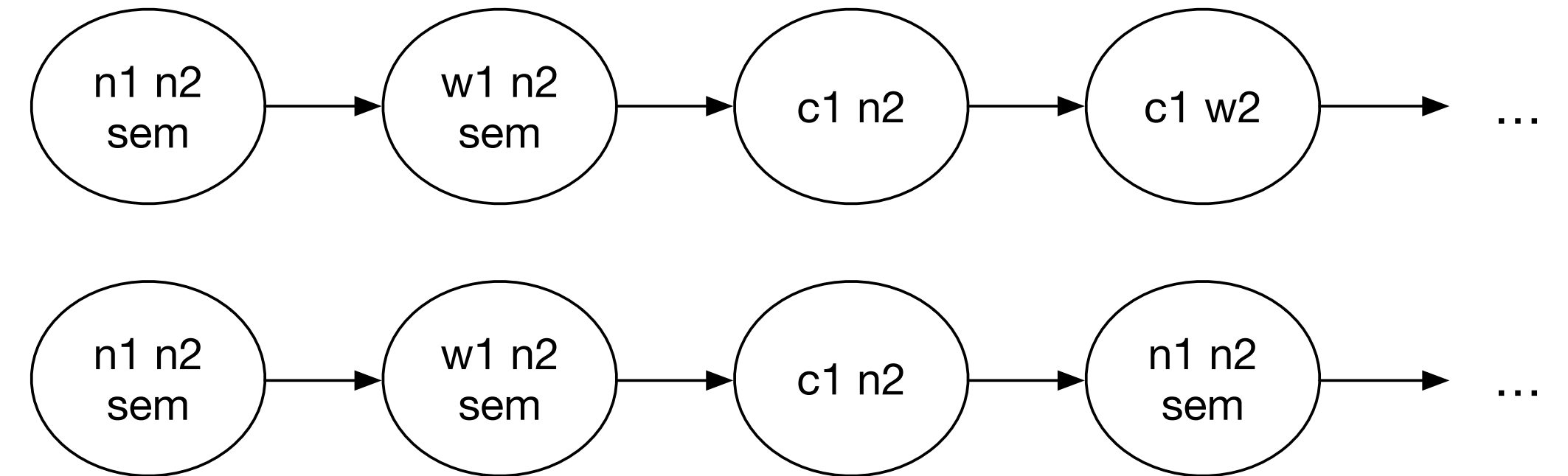
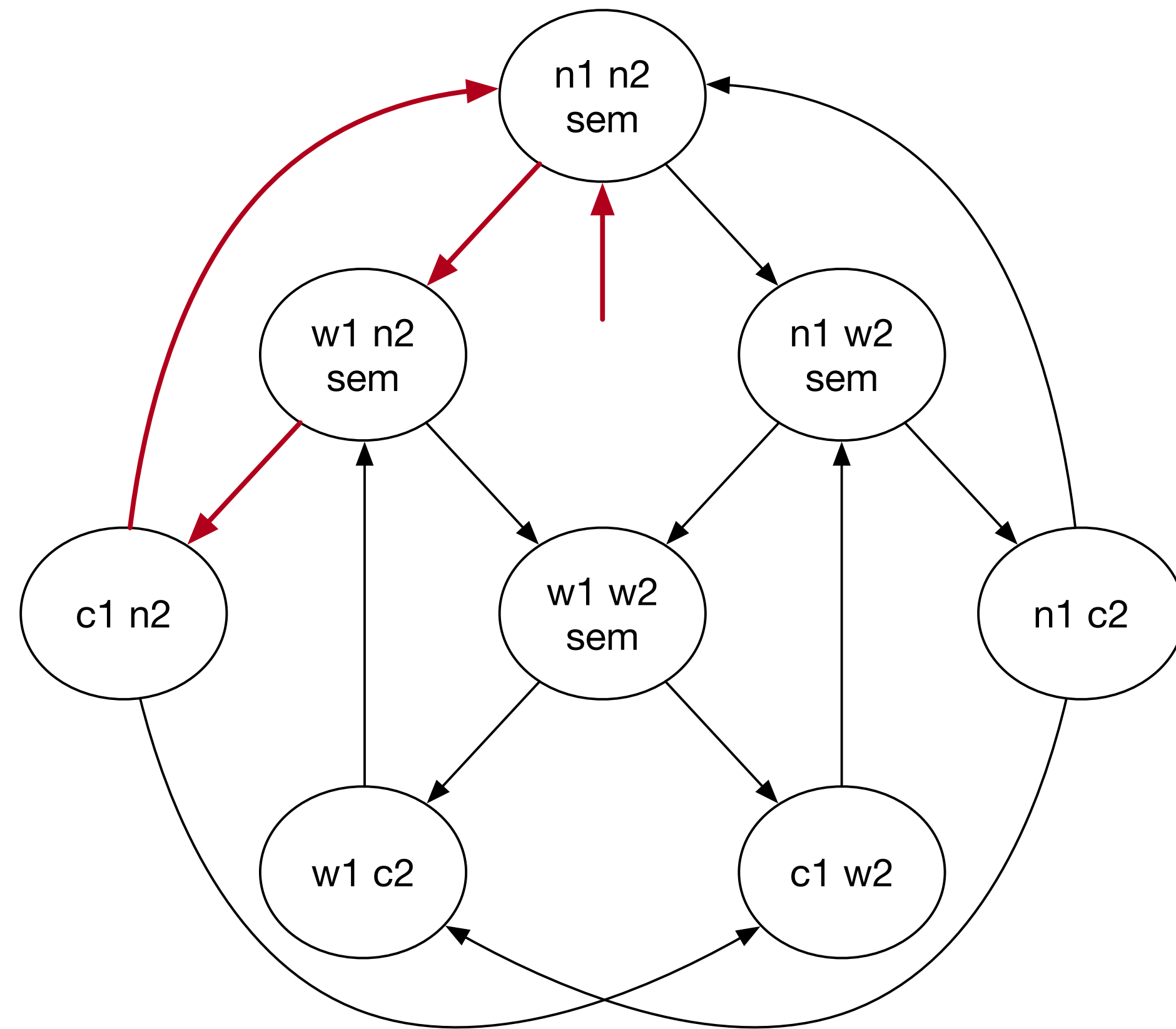
No starvation: $\text{AG } (w1 \rightarrow \text{AF } c1)$

Reversibility: $\text{AG EF } (n1 \wedge n2 \wedge \text{sem})$

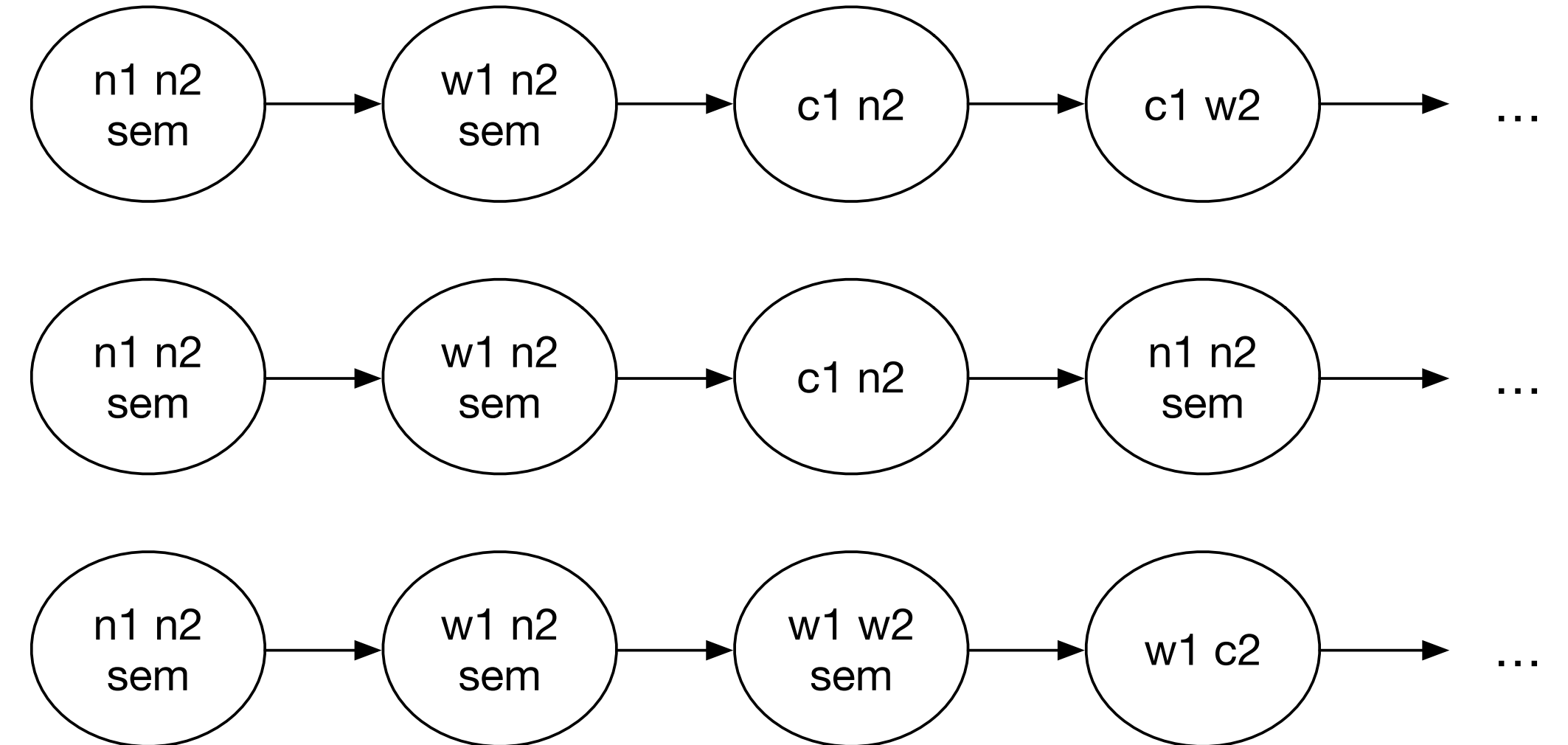
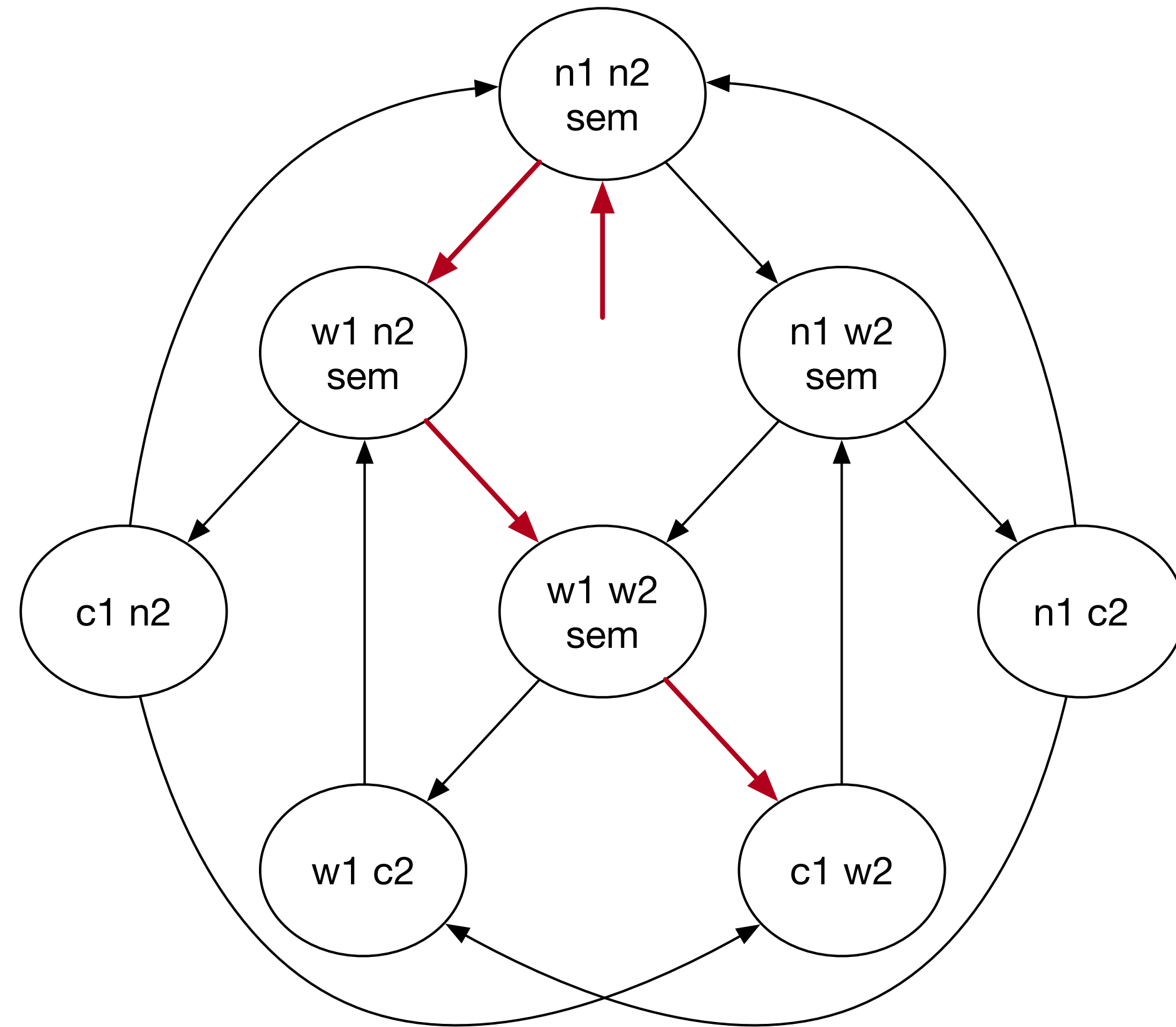
Linear Time



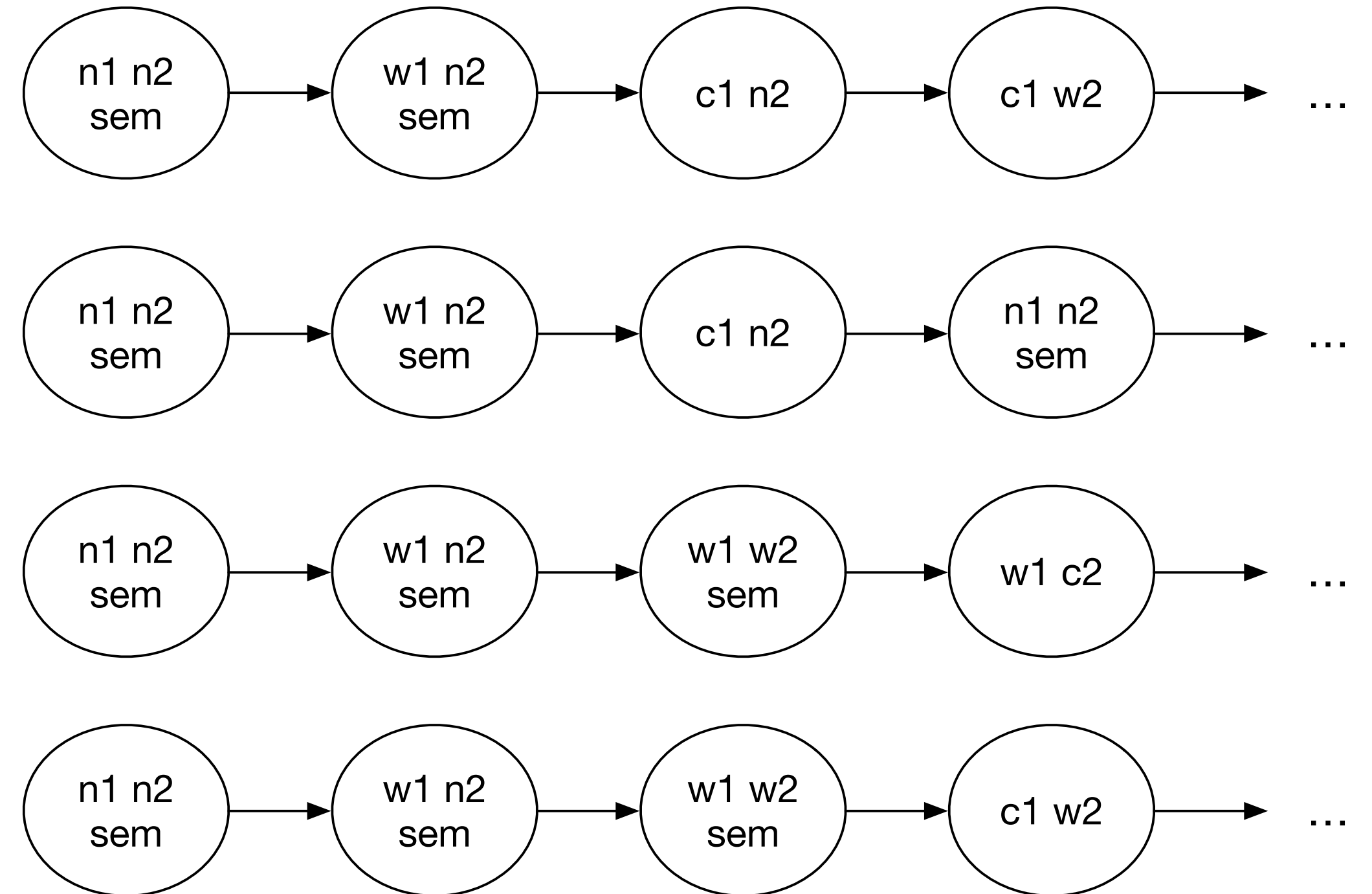
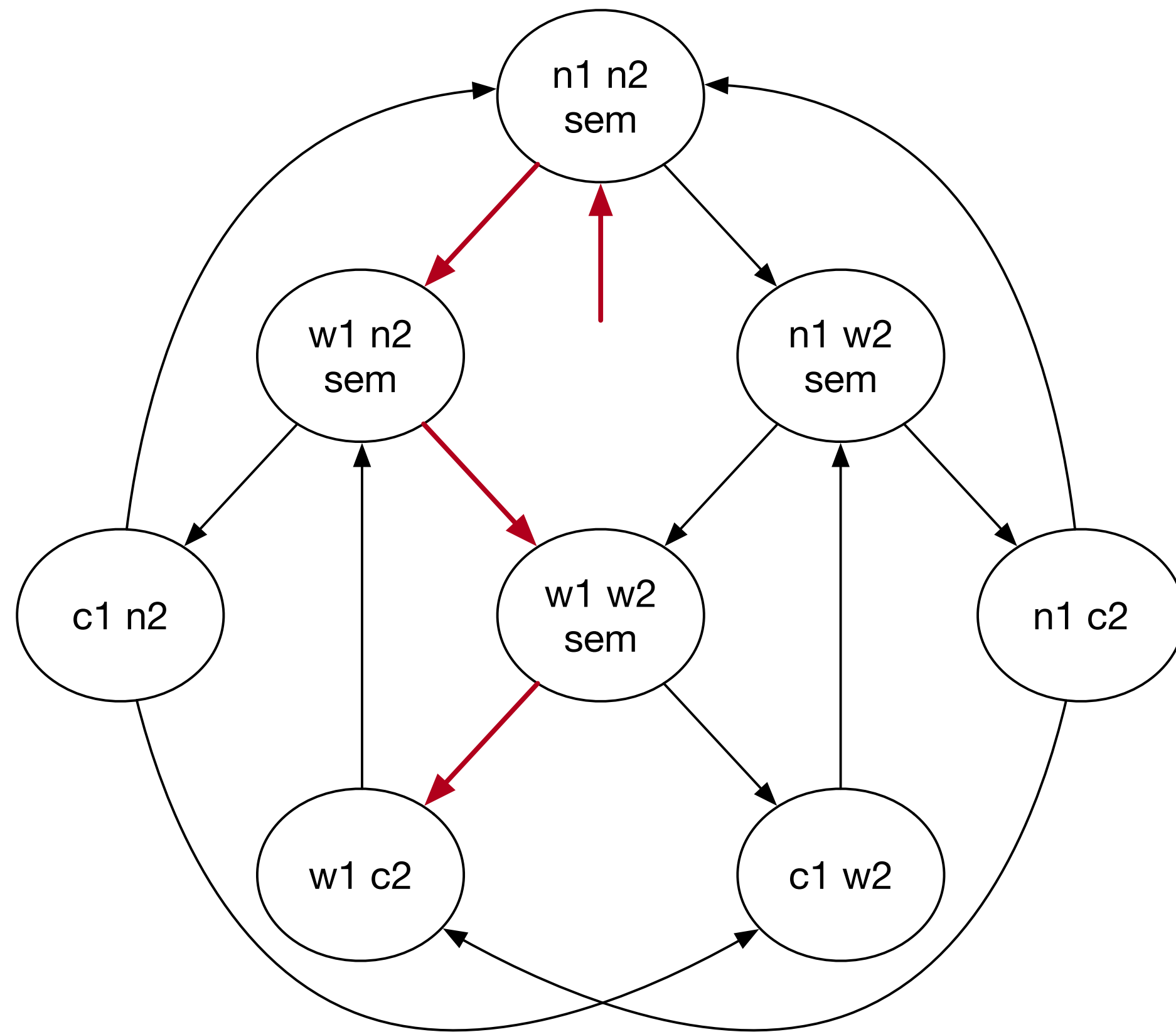
Linear Time



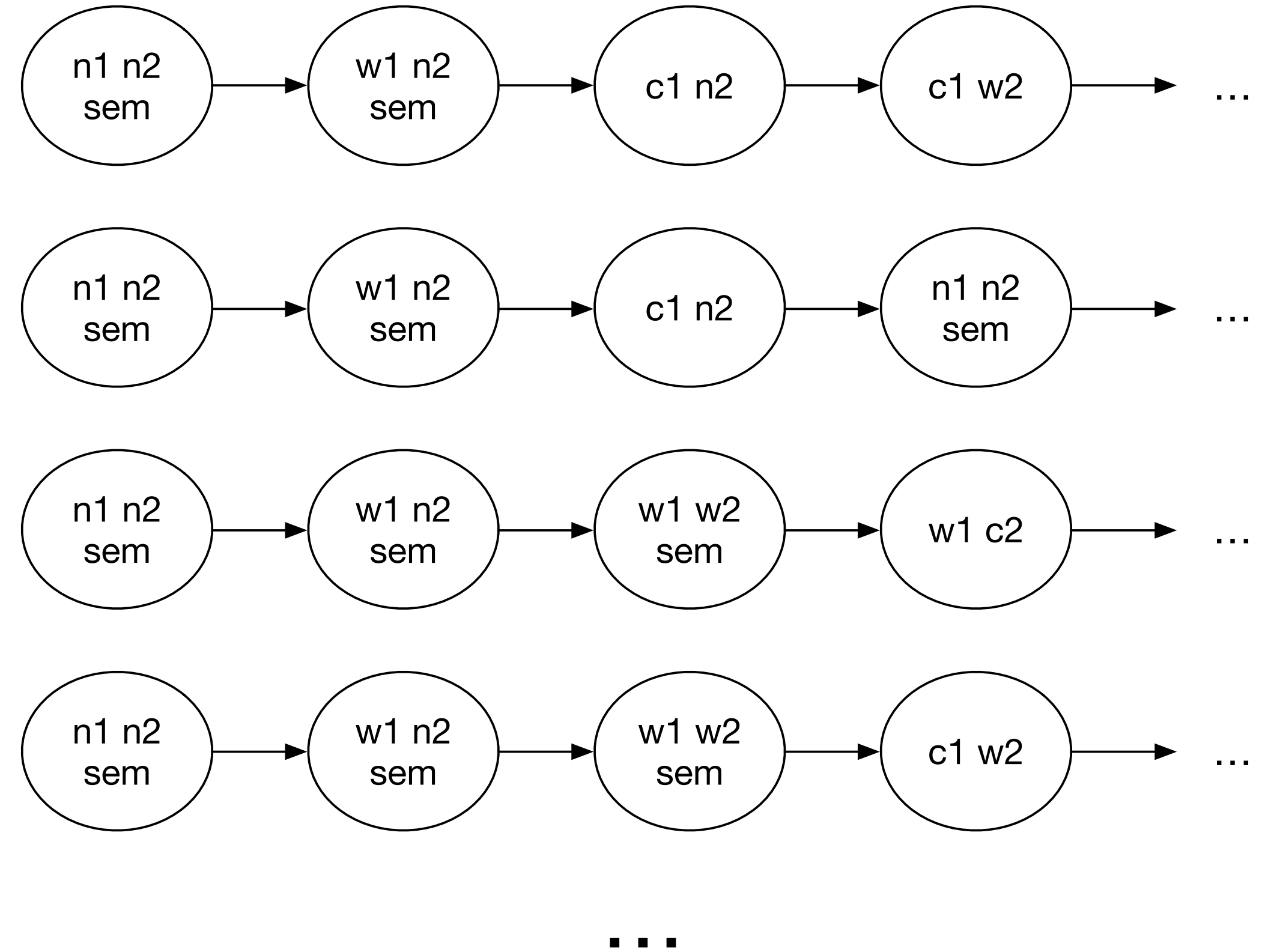
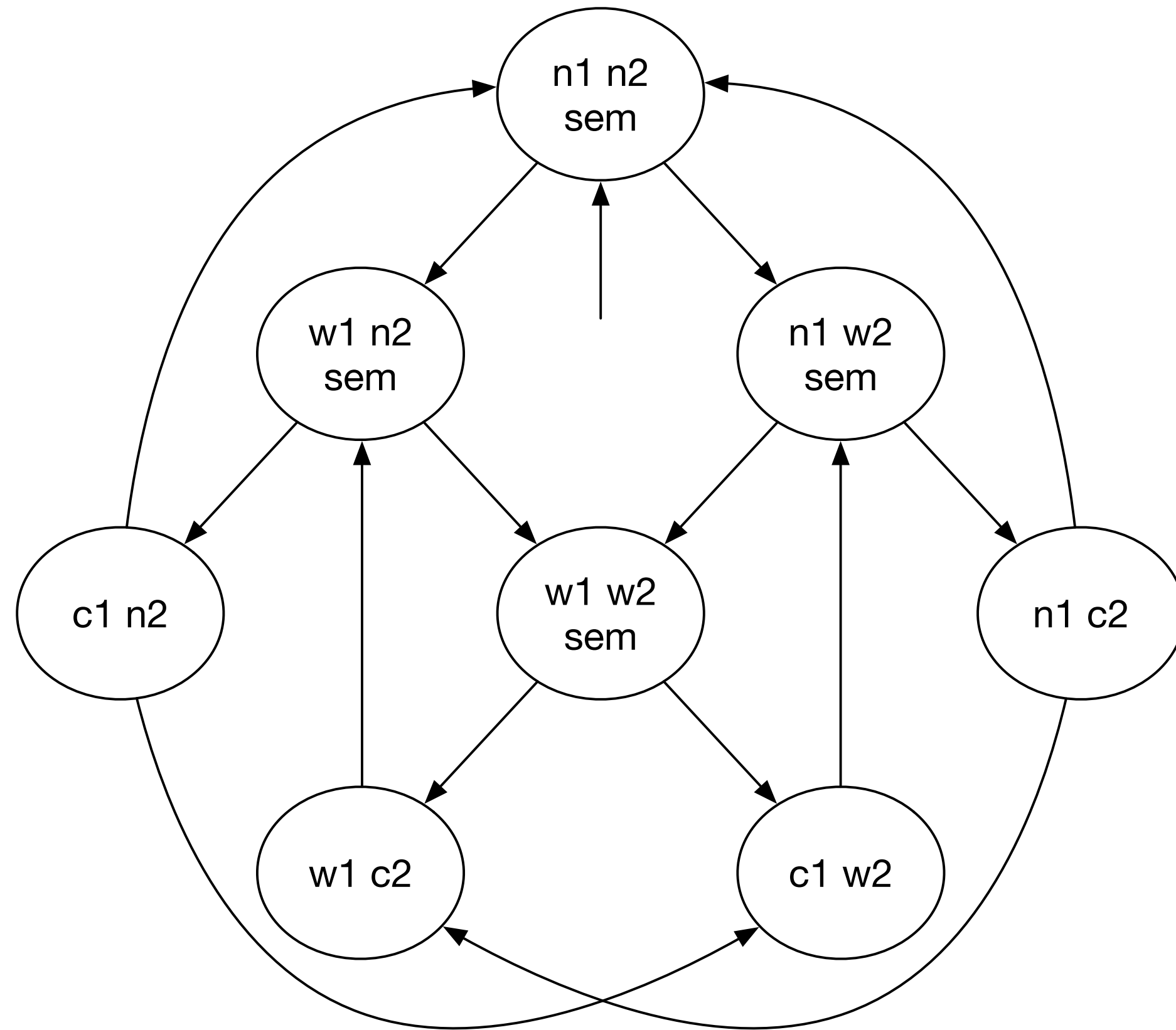
Linear Time



Linear Time



Linear Time



LTL

- **Linear Temporal Logic** (LTL) is a linear time logic
- LTL only has temporal operators

LTL Syntax

- **X** (or \circ), **G** (or \square), **F** (or \diamond), **U** and **R** are temporal operators:
 - **X** ϕ ϕ holds in the ne**X**t state
 - **G** ϕ ϕ always (or **G**lobally) holds
 - **F** ϕ ϕ eventually (or in the **F**uture) holds
 - ϕ **U** ψ ψ eventually holds and ϕ holds **U**ntil then
 - ϕ **R** ψ ψ always holds or until and ϕ holds (or **R**eleases)

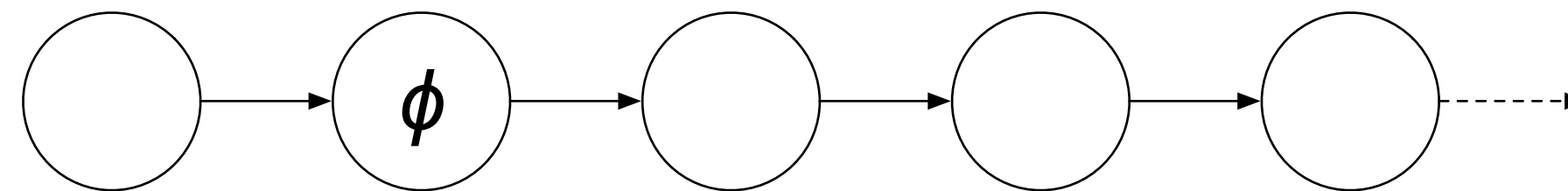
LTL Syntax

$$\begin{aligned} \phi, \psi ::= & \mathbf{X} \phi \mid \mathbf{G} \phi \mid \mathbf{F} \phi \\ & \mid \phi \mathbf{U} \psi \mid \phi \mathbf{R} \psi \\ & \mid p \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \neg \phi \mid \top \mid \perp \end{aligned}$$

with $p \in P$ atomic propositions

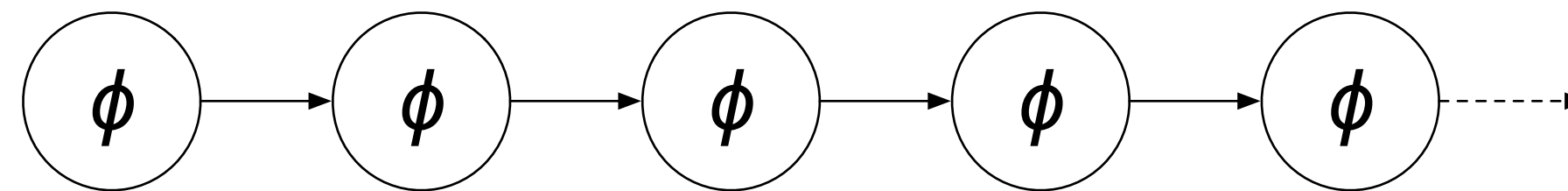
LTL Semantics

$X \phi$



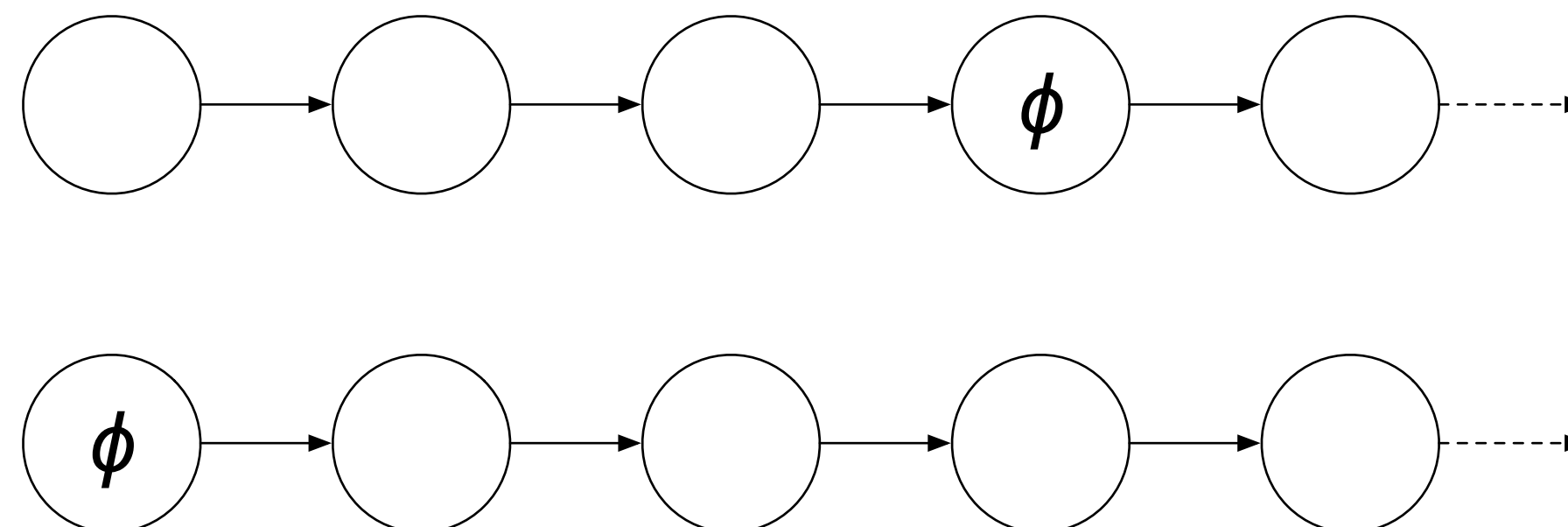
LTL Semantics

G ϕ



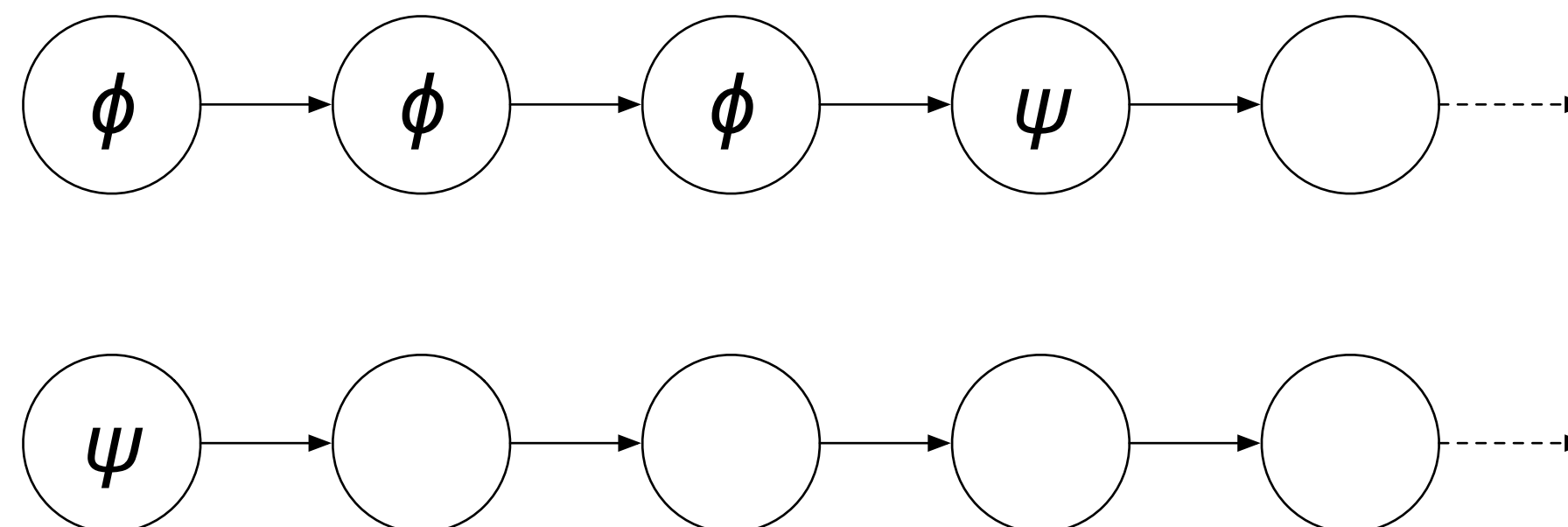
LTL Semantics

$\mathbf{F} \phi$



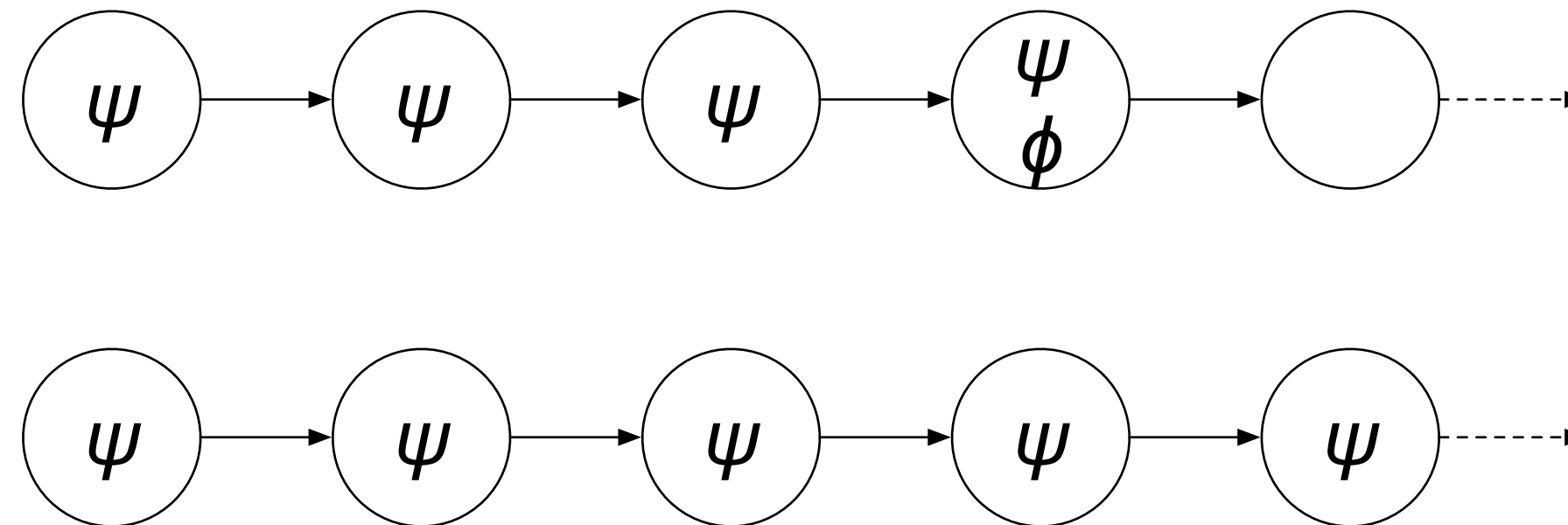
LTL Semantics

$\phi \mathbf{U} \psi$



LTL Semantics

$$\phi R \psi$$



LTL Semantics

- If an LTL formula ϕ holds for a Kripke structure $M = (S, I, R, L)$ we say

$$M \models \phi$$

- $M \models \phi$ iff for all paths $\pi \in M$ such that $\pi_0 \in I$ we have $M, \pi \models \phi$
- Minimal LTL subset: \top , \vee , \neg , **U**, **X**
 - $\mathbf{F}\phi \quad \equiv \quad \top \mathbf{U} \phi$
 - $\mathbf{G}\phi \quad \equiv \quad \neg \mathbf{F}(\neg \phi)$
 - $\phi \mathbf{R} \psi \quad \equiv \quad \neg((\neg \phi) \mathbf{U} (\neg \psi))$

LTL Semantics

$$M, \pi \models p \quad \equiv \quad p \in L(\pi_0)$$

$$M, \pi \models \top \quad \equiv \quad \top$$

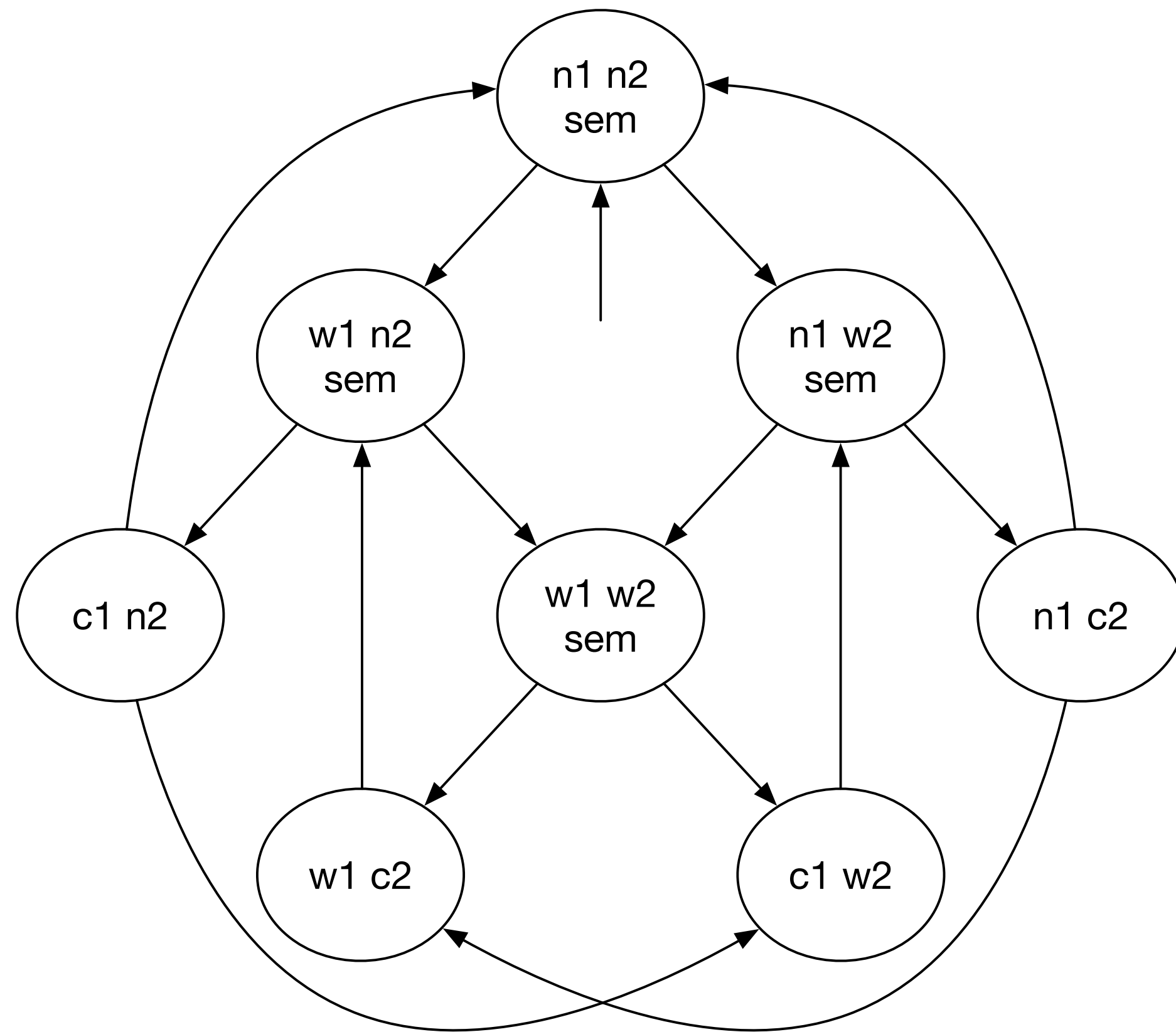
$$M, \pi \models \neg \phi \quad \equiv \quad M, \pi \not\models \phi$$

$$M, \pi \models \phi \vee \psi \quad \equiv \quad M, \pi \models \phi \text{ or } M, \pi \models \psi$$

$$M, \pi \models \mathbf{X}\phi \quad \equiv \quad M, \pi^1 \models \phi$$

$$M, \pi \models \phi \mathbf{U} \psi \quad \equiv \quad \exists i \geq 0 . (M, \pi^i \models \psi \text{ and } \forall 0 \leq j < i \ M, \pi^j \models \phi)$$

LTL Examples



Mutual exclusion: $\mathbf{G} \neg(c1 \wedge c2)$

No starvation: $\mathbf{G} (w1 \rightarrow \mathbf{F} c1)$

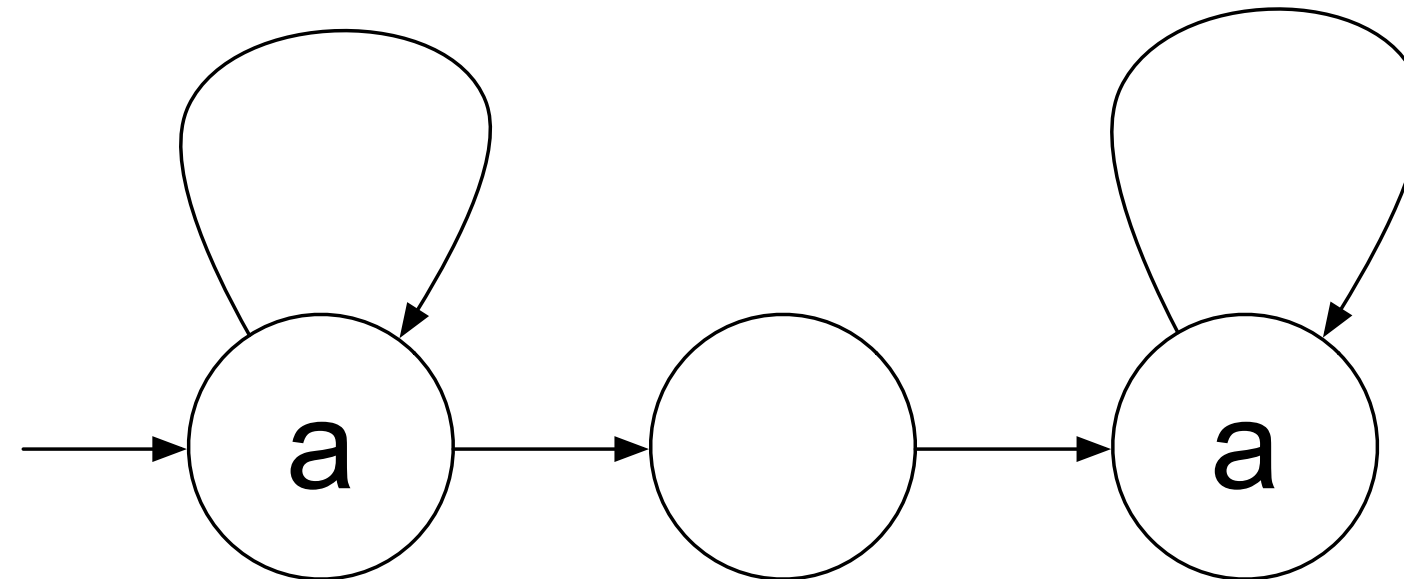
Reversibility: *NA*

LTL vs. CTL

- Many properties are expressible both in LTL and CTL
- LTL tends to be more intuitive, but CTL model checking is more efficient
- However, the two logics are **incomparable**: there are LTL properties not expressible in CTL, and vice-versa

LTL vs. CTL

- The CTL property **AG EF** ϕ is not expressible in LTL
- The LTL property **FG** ϕ (needed for *fairness*) is not expressible in CTL
- In general it is not sufficient to just add **A** quantifiers, e.g., **AF AG** a :



Model Checking Algorithms

- Two approaches to model checking:
 - **Complete:** considers paths of arbitrary length
 - **Bounded:** considers paths up to a certain length
- The latter may miss certain counter-examples but is more efficient
- We will focus on algorithms for model checking LTL

LTL Complete Model Checking

- The common approach is **automata-based**
- Create an automaton A_M whose language are the paths acceptable by M

$$L(A_M) = \{ w \mid w \in M \}$$

- Create an automaton A_ϕ whose language are the paths valid under ϕ

$$L(A_\phi) = \{ \pi \mid \pi \models \phi \}$$

- $M \models \phi$ iff $L(A_M) \subseteq L(A_\phi)$, or, more easily checked, iff

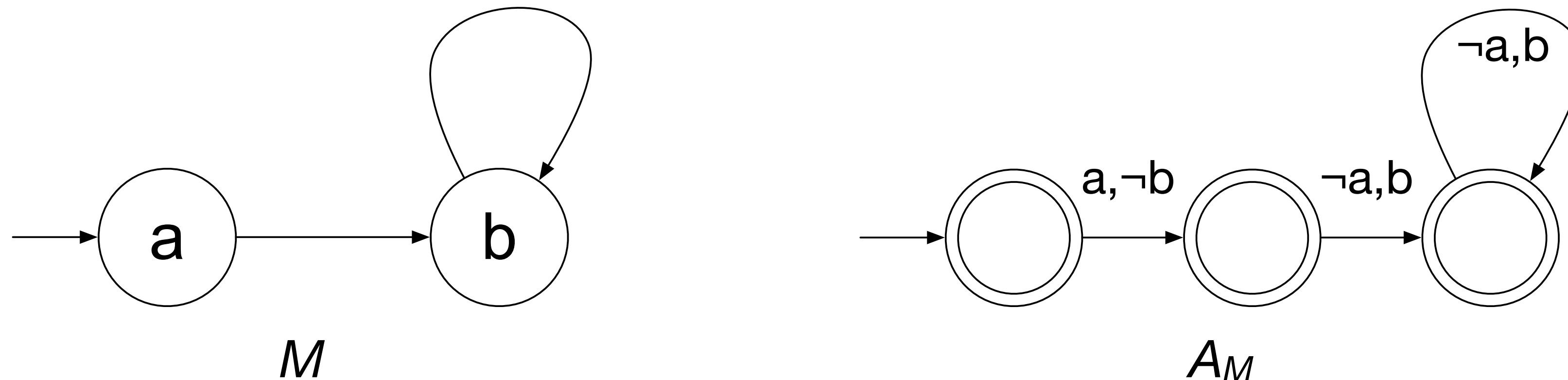
$$L(A_M) \cap L(A_{\neg\phi}) = \emptyset$$

Büchi Automata

- Finite-state automata cannot process infinite computations
- We need the notion of non-deterministic **Büchi automata** (NBA)
- An NBA $A = (Q, \Sigma, \delta, q_0, F)$ is:
 - Q is the set of states
 - Σ is the alphabet
 - $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function
 - $q_0 \in Q$ is the initial state
 - $F \subseteq Q$ is the set of accepting states
- A run is accepted by A if an accepting state is visited **infinitely often**

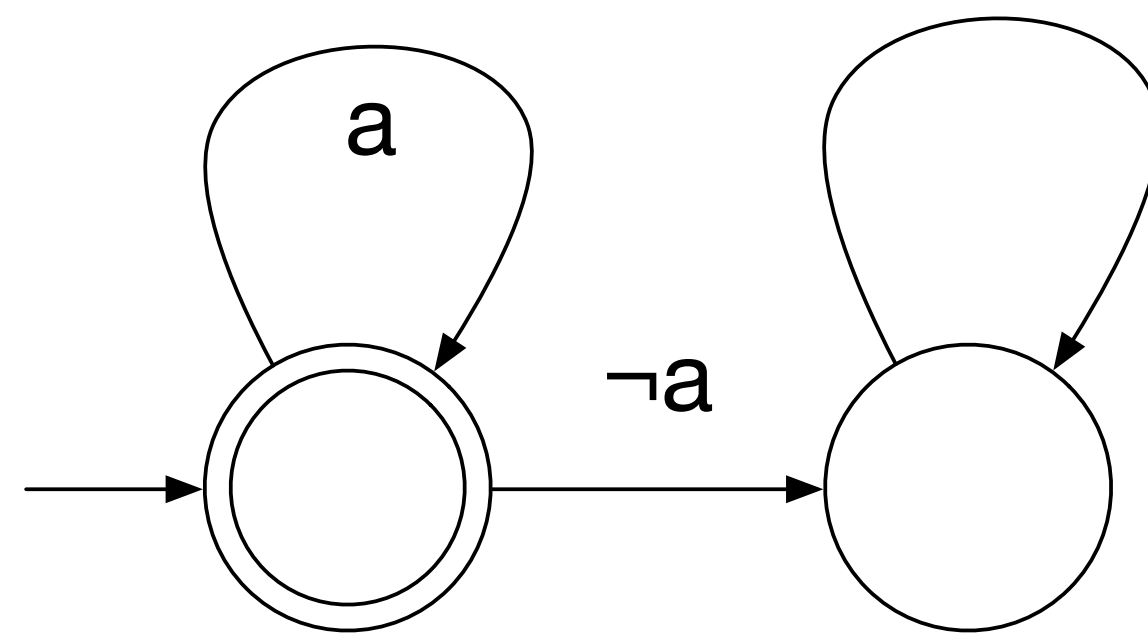
From M to Büchi Automata

- Construct A_M from M :
 - $\Sigma = 2^P$, conjunctions of atomic propositions
 - Add an initial state to the existing Kripke structure M
 - $s_1 \in \delta(s_2, w)$ iff in M $(s_1, s_2) \in R$ and $L(s_2) = w$
 - $F = Q$, all states are accepting

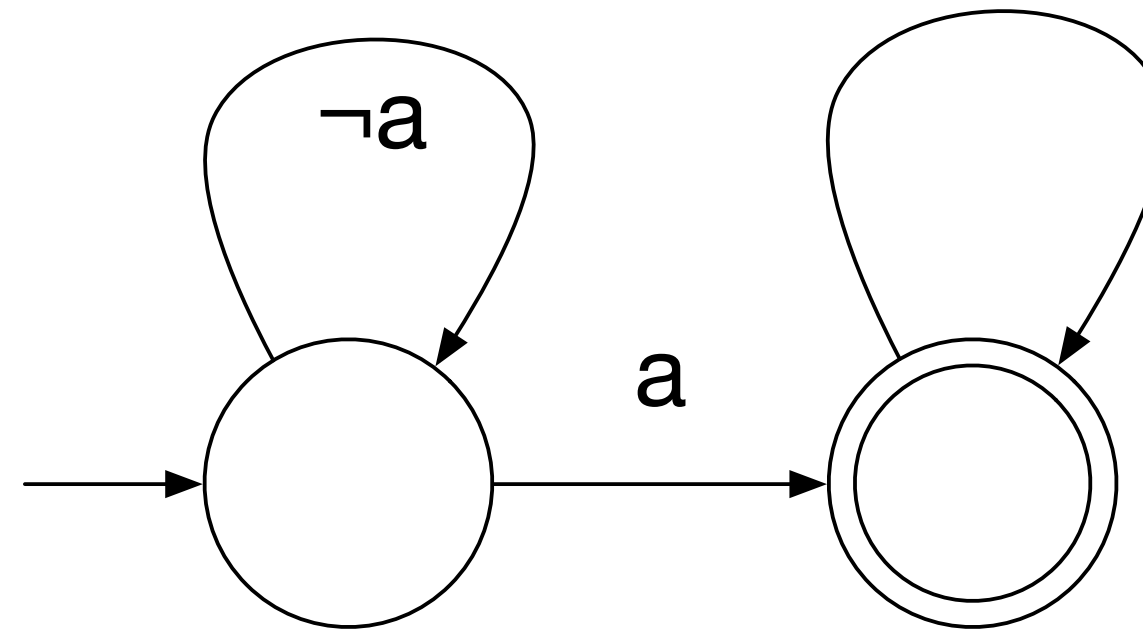


From ϕ to Büchi Automata

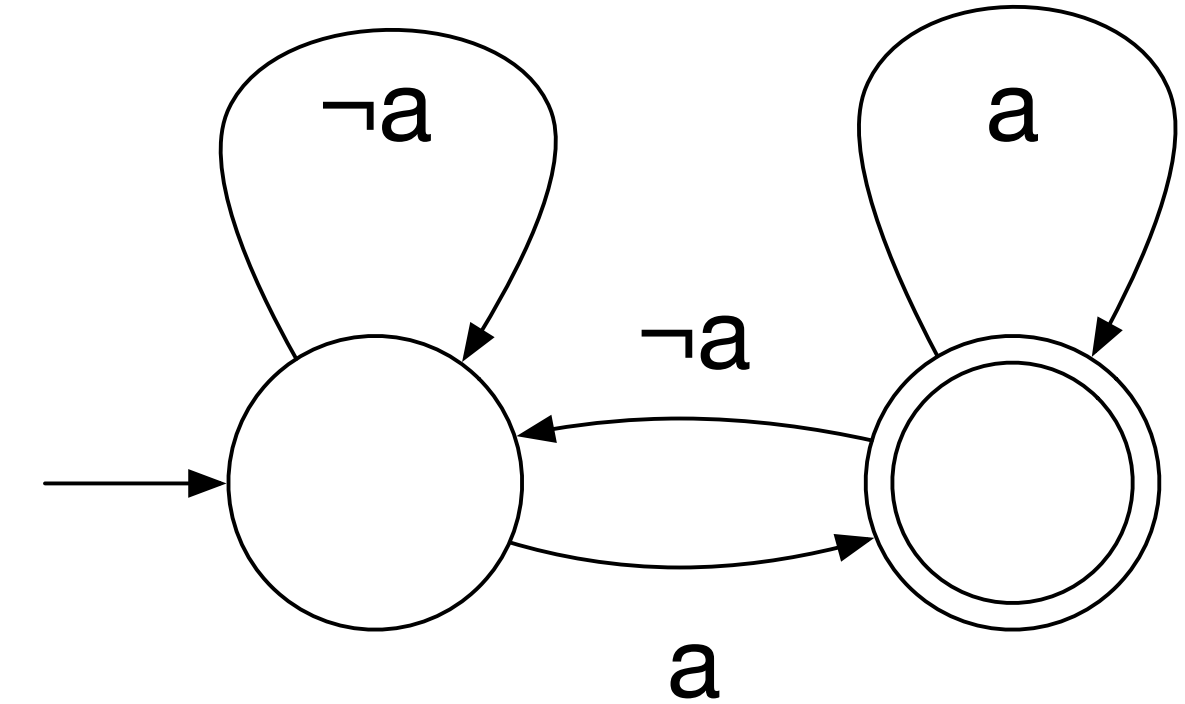
- Construct A_ϕ from ϕ :
 - $\Sigma = 2^P$, conjunctions of atomic propositions
 - Inductively create the states and transitions from ϕ (process omitted)



Ga



Fa



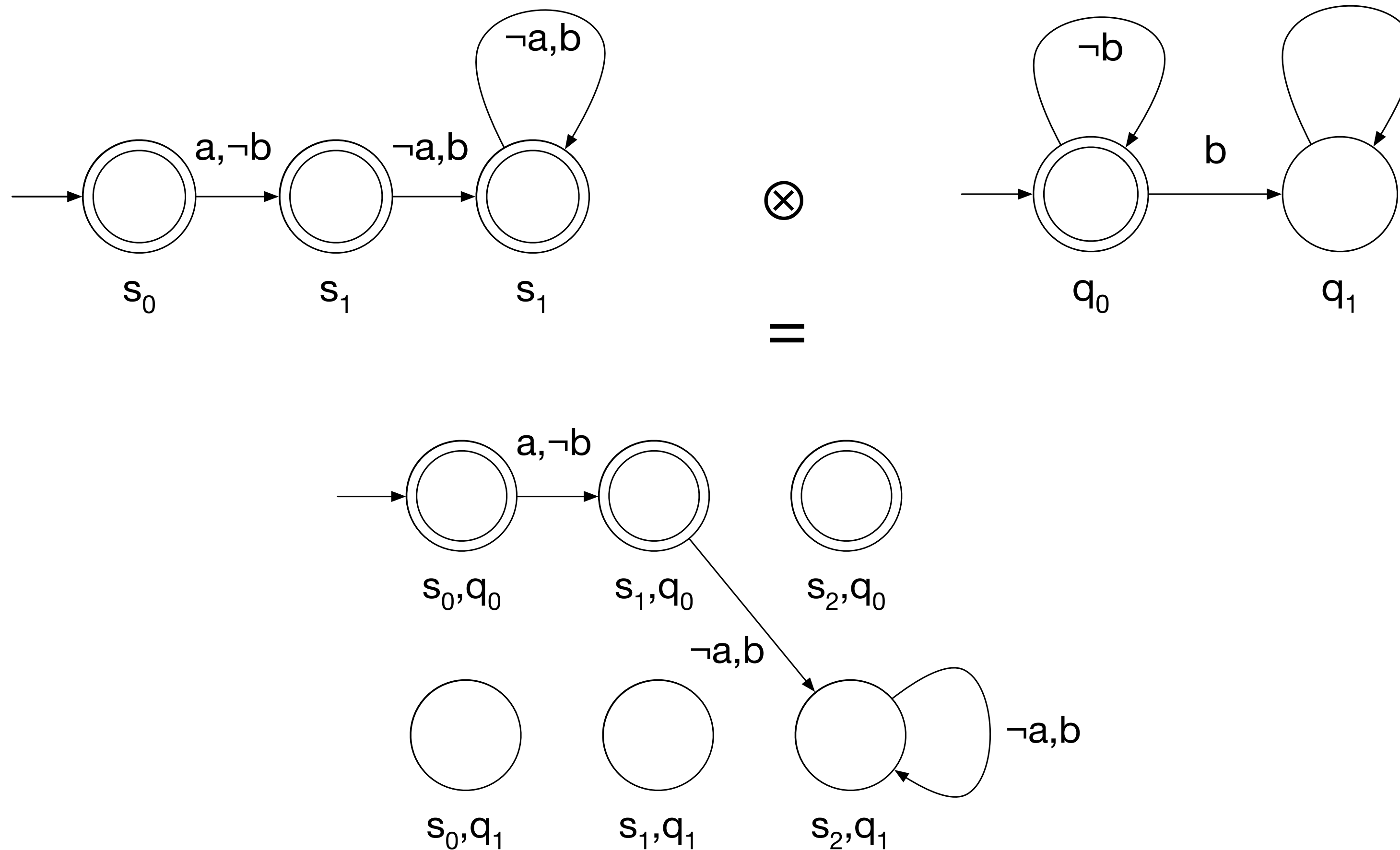
GFa

Checking for emptiness

- To test whether languages are disjoint, check whether the product automaton is empty
 - $M \models \phi$ iff $L(A_M) \cap L(A_\phi) = \emptyset$ iff $L(A_M \otimes A_\phi) = \emptyset$
- For $A_M = (Q_M, \Sigma, \delta_M, q_{0M}, Q_M)$ and $A_{\neg\phi} = (Q_{\neg\phi}, \Sigma, \delta_{\neg\phi}, q_{0\neg\phi}, F_{\neg\phi})$ we have
 - $A_M \otimes A_\phi = (Q_M \times Q_{\neg\phi}, \Sigma, \delta, q_{0M} \times q_{0\neg\phi}, Q_M \times F_{\neg\phi})$
 - $(s_2, q_2) \in \delta((s_1, q_1), w)$ iff $s_2 \in \delta_M(s_1, w)$ and $q_2 \in \delta_{\neg\phi}(q_1, w)$

Checking for emptiness

$$M \models \mathbf{F}b \quad \text{iff} \quad L(A_M \otimes A_{\mathbf{G}\neg b}) = \emptyset$$



Checking for emptiness

- Compute Strongly Connected Components (SCC) and check if there is one with accepting state (reachable from initial state)
 - Requires creating the complete automaton
- Determine reachable states with Depth-First Search (DFS), and then do a nested DFS to check whether there is a cycle
 - Can be performed on-the-fly
- Use a **fair** CTL model checker and test for **EG** \top
 - Benefits from advances in CTL model checkers (e.g., symbolic)

LTL Bounded Model Checking

- Checking a property for paths with a fixed length k simplifies the problem
 - We know exactly the size of the counter-examples ($k \times$ size of each state), so **SAT solvers** can be used
- To be sound, process should consider prefixes with **lassos**
- Some optimisations may ignore lassos, but adapted semantics:
 - **X** ϕ false in last state
 - **G** ϕ always false
 - for **F** ϕ , ϕ must occur in prefix

SAT Solving

- Procedures to solve the **Boolean satisfiability** problem (SAT)
- Is there a valuation for a set of Boolean variables that makes a proposition true?
- Very low-level, problems in **conjunctive normal form** (CNF)
- Constant advances, techniques often use such solvers in the backend

From M to SAT

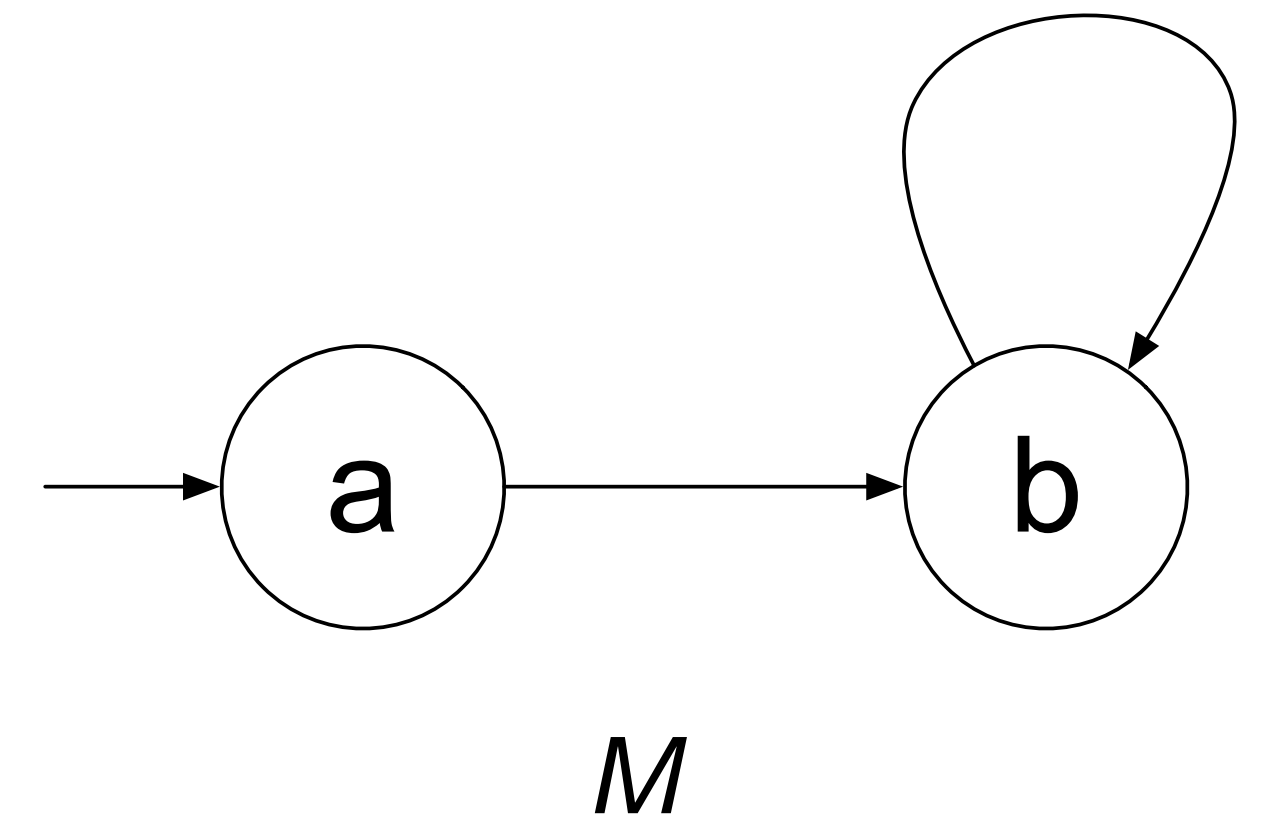
- For a length k
 - For each atomic proposition in $p \in P$ and each $0 \leq i < k$, create a Boolean variable p_i
 - Create a variable l_i for $0 \leq i < k$, to represent possible lassos
 - Let $I(s)$ be a predicate that holds when s is initial and $T(s_1, s_2)$ when there is a transition between s_1 and s_2 in R .
 - M can be encoded for a lasso l as:

$$[M]_l \equiv I(s_0) \wedge \bigwedge_{i=0}^{k-2} T(s_i, s_{i+1}) \wedge T(s_{k-1}, s_l)$$

From M to SAT

- For $k = 3$, variables are $a_0, a_1, a_2, b_0, b_1, b_2$ (since $P = \{a, b\}$)
- $I(s_0) = a_0 \wedge \neg b_0$
- $T(s_i, s_{i+1}) = (a_i \wedge \neg b_i \wedge \neg a_{i+1} \wedge b_{i+1}) \vee (\neg a_i \wedge b_i \wedge \neg a_{i+1} \wedge b_{i+1})$
- For a particular lasso l , $[M]_l$ will be

$$\begin{aligned}
 & (a_0 \wedge \neg b_0) \\
 & \quad \wedge \\
 & ((a_0 \wedge \neg b_0 \wedge \neg a_1 \wedge b_1) \vee (\neg a_0 \wedge b_0 \wedge \neg a_1 \wedge b_1)) \\
 & \quad \wedge \\
 & ((a_1 \wedge \neg b_1 \wedge \neg a_2 \wedge b_2) \vee (\neg a_1 \wedge b_1 \wedge \neg a_2 \wedge b_2)) \\
 & \quad \wedge \\
 & ((a_2 \wedge \neg b_2 \wedge \neg a_l \wedge b_l) \vee (\neg a_2 \wedge b_2 \wedge \neg a_l \wedge b_l))
 \end{aligned}$$



From ϕ to SAT

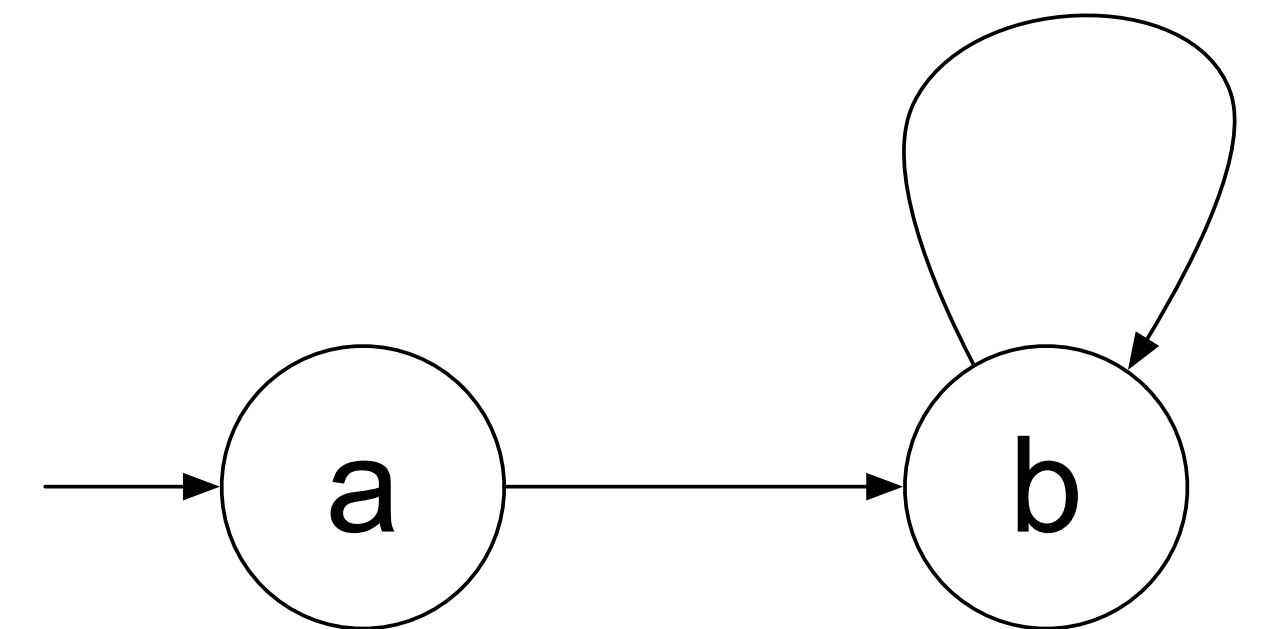
- For a length k , particular lasso l , translate ϕ for a state i as:
 - $i[p]_l = p_i$
 - $i[\neg p]_l = \neg p_i$
 - $i[\phi \vee \psi]_l = i[\phi]_l \vee i[\psi]_l$
 - $i[\mathbf{G}\phi]_l = i[\phi]_l \wedge \text{succ}(i)[\mathbf{G}\phi]_l$
 - $i[\mathbf{F}\phi]_l = i[\phi]_l \vee \text{succ}(i)[\mathbf{F}\phi]_l$
 - $i[\phi \mathbf{U} \psi]_l = i[\psi]_l \vee (i[\phi]_l \wedge \text{succ}(i)[\phi \mathbf{U} \psi]_l)$
- where $\text{succ}(i) = i+1$ if $l < k$, and l otherwise

From ϕ to SAT

- For $k = 3$ and a particular lasso l , $_0[\mathbf{G}(b \rightarrow \mathbf{X}a)]_l$ will be

$$(\neg b_0 \vee a_1) \wedge (\neg b_1 \vee a_2) \wedge (\neg b_2 \vee a_l)$$

- Putting it all together, $M \models \phi$ would be translated as:



M

$$\bigvee_{l=0}^{l < k} ([M]_l \wedge _0[\neg \phi]_l)$$