Model Checking Abstracto

Herramientas Avanzadas para el Desarrollo de Software

Profesora: Sonia Flores

Departamento de Informática,
Facultad de Matemáticas
UNSL

Curso 2007

Indice

Objetivos del tema

- Repasar las nociones básicas del *model checking*
- Aplicar la técnica de interpretación abstracta al *model checking*
- Conocer los aspectos fundamentales de preservación de resultados

Introducción

IDEA

Si conseguimos demostrar una propiedad en el universo abstracto, queremos garantizar que dicha propiedad se cumple también en el universo concreto

- todo concreto debe tener una descripción
- todo abstracto debe representar a algún concreto
- se deben preservar las propiedades
  - de forma débil
  - de forma fuerte

Introducción

IDEA

Si conseguimos demostrar una propiedad en el universo abstracto, queremos garantizar que dicha propiedad se cumple también en el universo concreto

- todo concreto debe tener una descripción
- todo abstracto debe representar a algún concreto
- se deben preservar las propiedades
  - de forma débil
  - de forma fuerte

Introducción

SISTEMAS

- Sistemas funcionales

- Sistemas reactivos

Introducción

SISTEMAS

- Sistemas funcionales
  - punto final concreto
  - verificación de postcondiciones (entrada/salida)
  - lógica de Hoare
  - aplicamos *testing*, depuración, ...
- Sistemas reactivos

Introducción

SISTEMAS

- Sistemas funcionales
  - punto final concreto
  - verificación de postcondiciones (entrada/salida)
  - lógica de Hoare
  - aplicamos *testing*, depuración, ...
- Sistemas reactivos
  - no tienen un *final*
  - sistemas que interactúan con el usuario (u otros sistemas)
  - especificados como sistemas concurrentes: difíciles de verificar a mano
  - se razona sobre trazas y no entrada/salida: nuevas lógicas
  - comprobar propiedades dinámicas de forma estática

Introducción

Propiedades de Sistemas Reactivos

- Propiedades de seguridad
- Propiedades de viveza

- Propiedades universales
- Propiedades existenciales

Las técnicas usadas para verificar propiedades de seguridad universales basadas en la búsqueda de contraejemplos son distintas de las usadas para analizar propiedades existenciales o de viveza.

Introducción

Propiedades de Sistemas Reactivos

- Propiedades de seguridad
- Propiedades de viveza

- Propiedades universales
- Propiedades existenciales

Las técnicas usadas para verificar propiedades de seguridad universales basadas en la búsqueda de contraejemplos son distintas de las usadas para analizar propiedades existenciales o de viveza.

Introducción

Propiedades de Sistemas Reactivos

- Propiedades de seguridad
- Propiedades de viveza

- Propiedades universales
- Propiedades existenciales

Las técnicas usadas para verificar propiedades de seguridad universales basadas en la búsqueda de contraejemplos son distintas de las usadas para analizar propiedades existenciales o de viveza.

Model Checking

Model Checking

La técnica del *model checking* consiste en la comprobación de que un sistema satisface una determinada propiedad

$$\mathcal{M} \models \phi$$

Model Checking

Model Checking

La técnica del *model checking* consiste en la comprobación de que un sistema satisface una determinada propiedad

$$\mathcal{M} \models \phi$$

$$\forall x \in \mathit{init}(\mathcal{M}), \mathcal{M}, x \models \phi$$

Model Checking

Model Checking

La técnica del *model checking* consiste en la comprobación de que un sistema satisface una determinada propiedad

$$\mathcal{M} \models \phi$$

$$\forall x \in \mathit{init}(\mathcal{M}), \mathcal{M}, x \models \phi$$

- técnica formal: base matemática
- respuesta segura
- en caso de respuesta negativa, da contraejemplo
- aplicable (a priori) a sistemas *finitos*

Model Checking

PROBLEMAS

- Problema de la explosión del espacio de búsqueda
  - Método simbólico
  - Método *on-the-fly*
  - Método composicional
  - Método basado en simulaciones y órdenes
  - Método basado en la interpretación abstracta
  - ...
- Soluciones *ad-hoc*
- Lenguajes con potencia expresiva muy limitada

Model Checking
El modelo

El sistema suele representarse mediante un sistema de transiciones, normalmente etiquetado

Estructura de Kripke

Tupla $\langle S, I, R, L \rangle$ donde

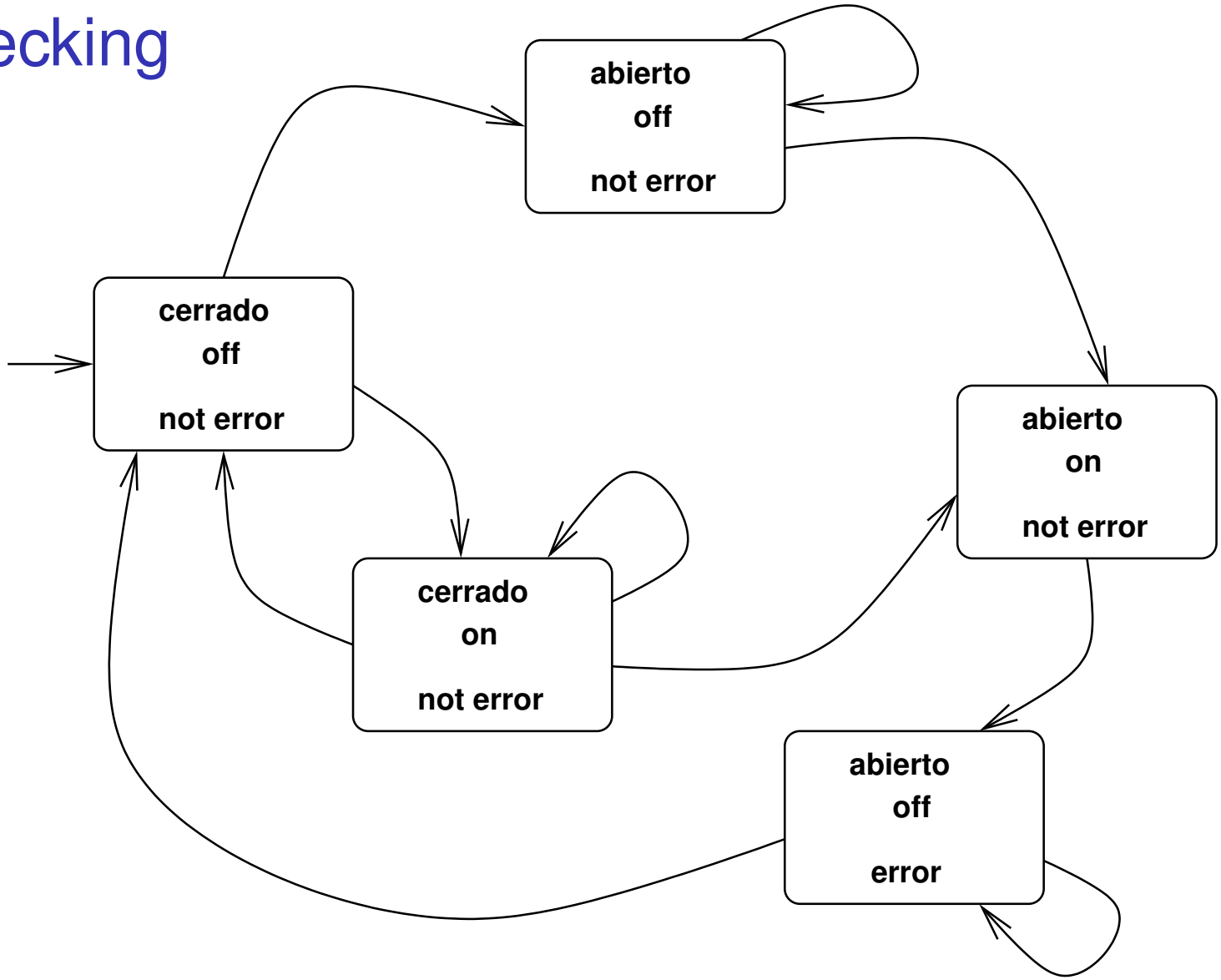
- S es un conjunto de estados
- $I \subseteq S$ es el conjunto de estados iniciales
- $R \subseteq S \times S$ es una relación (total) binaria
- $L : \wp(\Sigma) \rightarrow S$ es una función de etiquetado

Model Checking

Estructura de Kripke

- La estructura de *Kripke* tiene su origen en la interpretación de la lógica modal clásica
- Se suele exigir que la relación sea total para que todas las trazas del sistema sea infinitas, pero dependiendo del marco de trabajo, puede no ser necesaria esta condición
- Se representa de forma gráfica mediante un grafo dirigido

Model Checking



Model Checking
El modelo

Generación del modelo:

- podemos generarlo a mano
  - posibilidad de introducir errores adicionales
  - es necesario comprobar que el error se corresponde con el modelo real
- podemos generarlo a partir de otro modelo
  - aproximación basada en autómatas
- podemos generarlo a partir de una especificación de alto nivel
  - problema complejo: depende del lenguaje

Model Checking
El modelo

Generación del modelo:

- podemos generarlo a mano
  - posibilidad de introducir errores adicionales
  - es necesario comprobar que el error se corresponde con el modelo real
- podemos generarlo a partir de otro modelo
  - aproximación basada en autómatas
- podemos generarlo a partir de una especificación de alto nivel
  - problema complejo: depende del lenguaje

Model Checking
El modelo

Generación del modelo:

- podemos generarlo a mano
  - posibilidad de introducir errores adicionales
  - es necesario comprobar que el error se corresponde con el modelo real
- podemos generarlo a partir de otro modelo
  - aproximación basada en autómatas
- podemos generarlo a partir de una especificación de alto nivel
  - problema complejo: depende del lenguaje

Model Checking
La propiedad

Tradicionalmente, la propiedad se expresa en una lógica temporal

- La lógica temporal es un tipo de lógica modal
- Modalidades temporales: *siempre* y *eventualmente*
- Rescatadas de los libros en el 83 para aplicarse a la verificación de sistemas reactivos

Model Checking
Clasificación

Las lógicas temporales pueden clasificarse según distintos criterios:

- proposicionales *vs* de primer orden
- globales *vs* composicionales
- ramificadas *vs* lineales
- puntuales *vs* de intervalos
- discretas *vs* continuas (o densas)
- de pasado *vs* de futuro

Model Checking
La lógica CTL*

CTL*

La *Computational Tree Logic* es la versión enriquecida de la lógica ramificada CTL

- útil para razonar con sistemas no deterministas
- tiene operadores temporales
- tiene cuantificadores sobre caminos
  - puede considerar más de un posible futuro en cada instante de tiempo

Model Checking
La lógica CTL*

Dos tipos de fórmulas:

- *state*-fórmulas

  Los valores de verdad se definen sobre un estado determinado
- *path*-fórmulas

  Los valores de verdad consideran un (conjunto de) camino(s) en el modelo

Model Checking
La lógica CTL*

SINTAXIS

Las *state*-fórmulas se definen como:

(S1) las proposiciones atómicas son *state*-fórmulas

(S2) si ϕ y ψ son *state*-fórmulas, entonces $\phi \wedge \psi$ y $\neg\phi$ también son *state*-fórmulas

(S3) si ϕ es una *path*-fórmula, entonces $A\phi$ y $E\phi$ son *state*-fórmulas

Model Checking
La lógica CTL*

SINTAXIS

Las *path*-fórmulas se definen como:

(P1) las *state*-fórmulas son *path*-fórmulas

(P2) si ϕ y ψ son *path*-fórmulas, entonces $\phi \wedge \psi$ y $\neg\phi$ también son *path*-fórmulas

(P3) si ϕ y ψ son *path*-fórmulas, entonces $\bigcirc\phi$ y $\phi \mathcal{U} \psi$ son también *path*-fórmulas

Model Checking
La lógica CTL*

SINTAXIS

Las *path*-fórmulas se definen como:

(P1) las *state*-fórmulas son *path*-fórmulas

(P2) si ϕ y ψ son *path*-fórmulas, entonces $\phi \wedge \psi$ y $\neg\phi$ también son *path*-fórmulas

(P3) si ϕ y ψ son *path*-fórmulas, entonces $\bigcirc\phi$ y $\phi \mathcal{U} \psi$ son también *path*-fórmulas

Definición de fragmentos LTL y CTL en el boletín

Model Checking
La lógica CTL*

La semántica de una fórmula CTL* se da tradicionalmente en función de una estructura de *Kripke*

- Dada la estructura $K = (S, I, R, L)$
- Un camino completo es una secuencia infinita $s = s_0 s_1 \cdots s_n \cdots$ de forma que $\forall i, (s_i, s_{i+1}) \in R$
- s^i representa el sufijo $s_i s_{i+1} \cdots$ de la secuencia s

Model Checking
La lógica CTL*

SEMÁNTICA

$K, s_0 \models \phi$

La state-fórmula ϕ es cierta en el estado s_0 según el modelo K

$K, s \models \phi$

La path-fórmula ϕ se cumple en el camino s según el modelo K

Model Checking
La lógica CTL*

SEMÁNTICA

$$K, s_0 \models \phi$$

La state-fórmula ϕ es cierta en el estado s_0 según el modelo K

$$K, s \models \phi$$

La path-fórmula ϕ se cumple en el camino s según el modelo K

Model Checking
La lógica CTL*

SEMÁNTICA (1/2)

$\models$ se define de forma inductiva a partir de las reglas:

(S1) $M, s_0 \models \phi$ sii $\phi \in L(s_0)$

(S2) $M, s_0 \models \phi \wedge \psi$ sii $M, s_0 \models \phi$ y además $M, s_0 \models \psi$

(S3) $M, s_0 \models \neg\phi$ sii $M, s_0 \not\models \phi$

(S4) $M, s_0 \models E\phi$ sii existe un *fullpath* $s = s_0, s_1, \ldots$ en M tal que $M, s \models \phi$

(S5) $M, s_0 \models A\phi$ sii para todo *fullpath* $s = s_0, s_1, \ldots$ en M tal que $M, s \models \phi$

Model Checking
La lógica CTL*

SEMÁNTICA (2/2)

(P1) $M, s \models \phi$ sii $M, s_0 \models \phi$

(P2) $M, s \models \phi \wedge \psi$ sii $M, s \models \phi$ y además $M, s \models \psi$

(P3) $M, s \models \neg\phi$ sii $M, s \not\models \phi$

(P4) $M, s \models \phi \mathcal{U} \psi$ sii existe un i tal que $M, s^i \models \psi$
y para todo j menor que i, $M, s^j \models \phi$

(P5) $M, s \models \bigcirc\phi$ sii $M, s^1 \models \phi$

Model Checking
La lógica CTL*

Validez

Una *state*-fórmula (*path*-fórmula) es válida si para todo modelo M y estado s_i (camino s), $M, s_i \models \phi$ ($M, s \models \phi$)

Satisfacibilidad

Una *state*-fórmula (*path*-fórmula) es satisfacible si existe un modelo M y estado s_i (camino s), $M, s_i \models \phi$ ($M, s \models \phi$)

Model Checking
La lógica CTL*

Validez

Una *state*-fórmula (*path*-fórmula) es válida si para todo modelo M y estado s_i (camino s), $M, s_i \models \phi$ ($M, s \models \phi$)

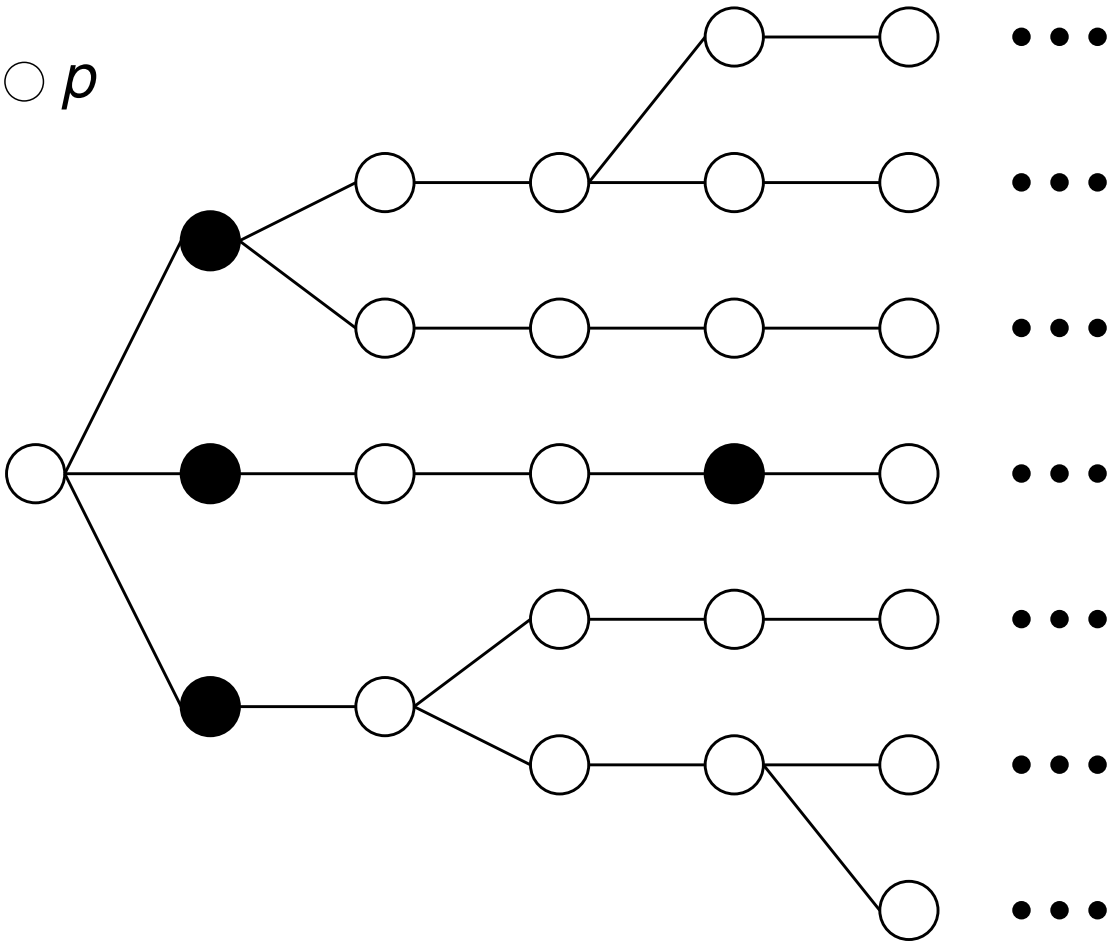
Satisfacibilidad

Una *state*-fórmula (*path*-fórmula) es satisfacible si existe un modelo M y estado s_i (camino s), $M, s_i \models \phi$ ($M, s \models \phi$)

Model Checking

Ejemplos (1/5)

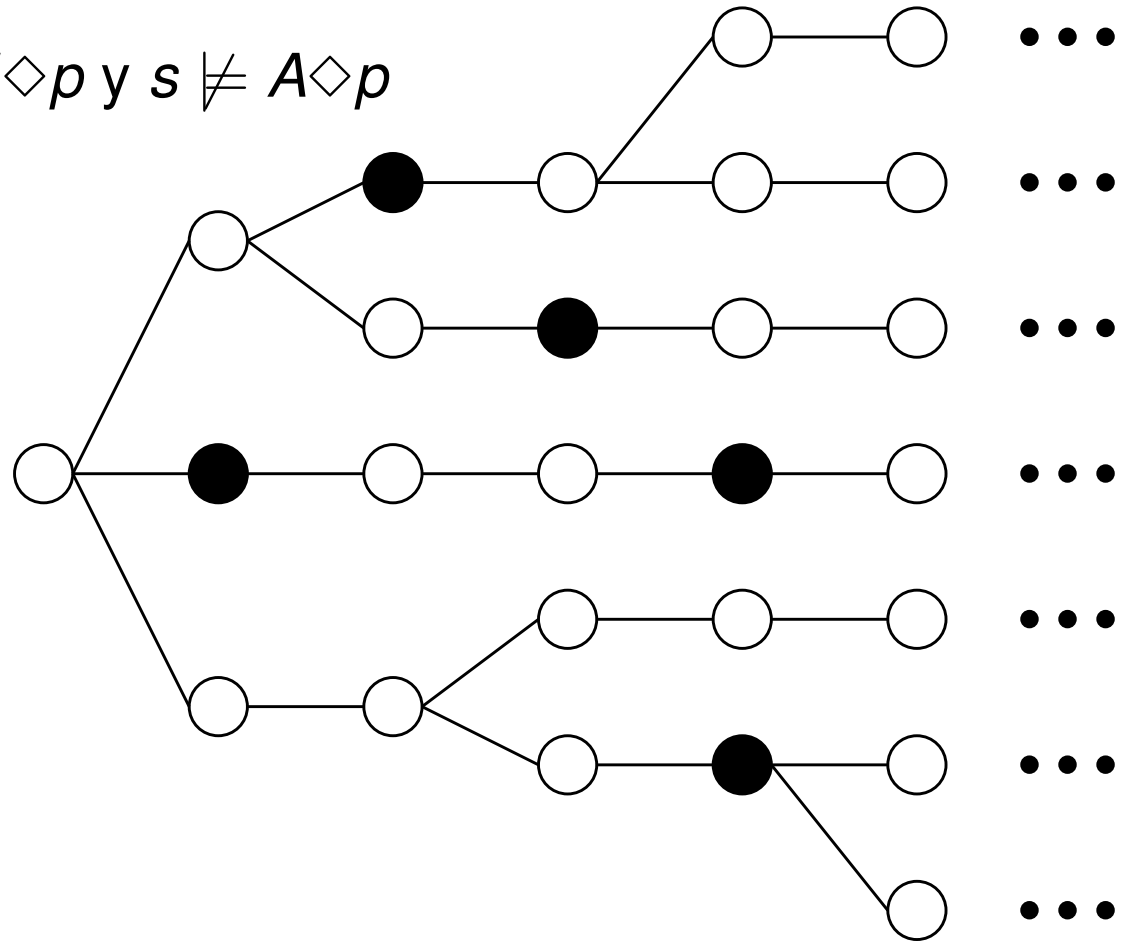
- $s \models A \bigcirc p$



Model Checking

Ejemplos (2/5)

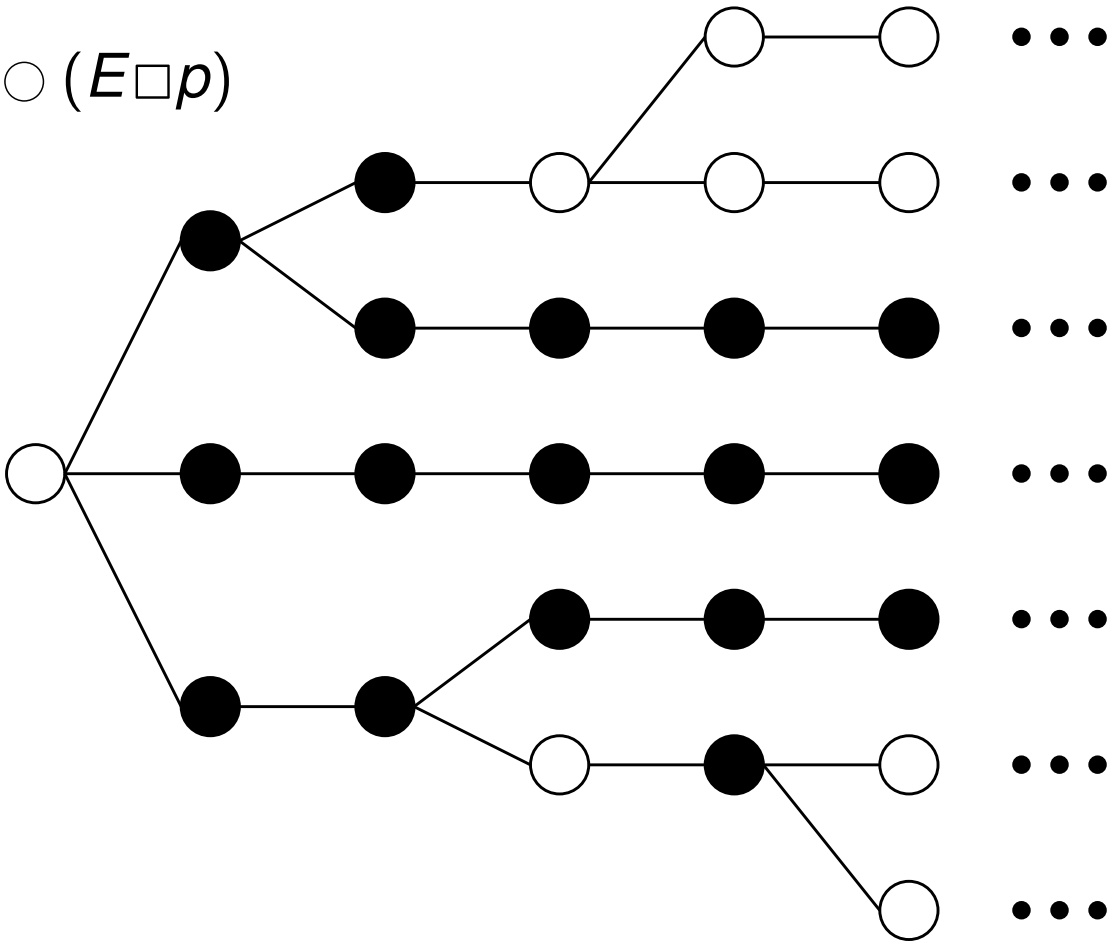
- $s \models E\Diamond p$ y $s \not\models A\Diamond p$



Model Checking

Ejemplos (3/5)

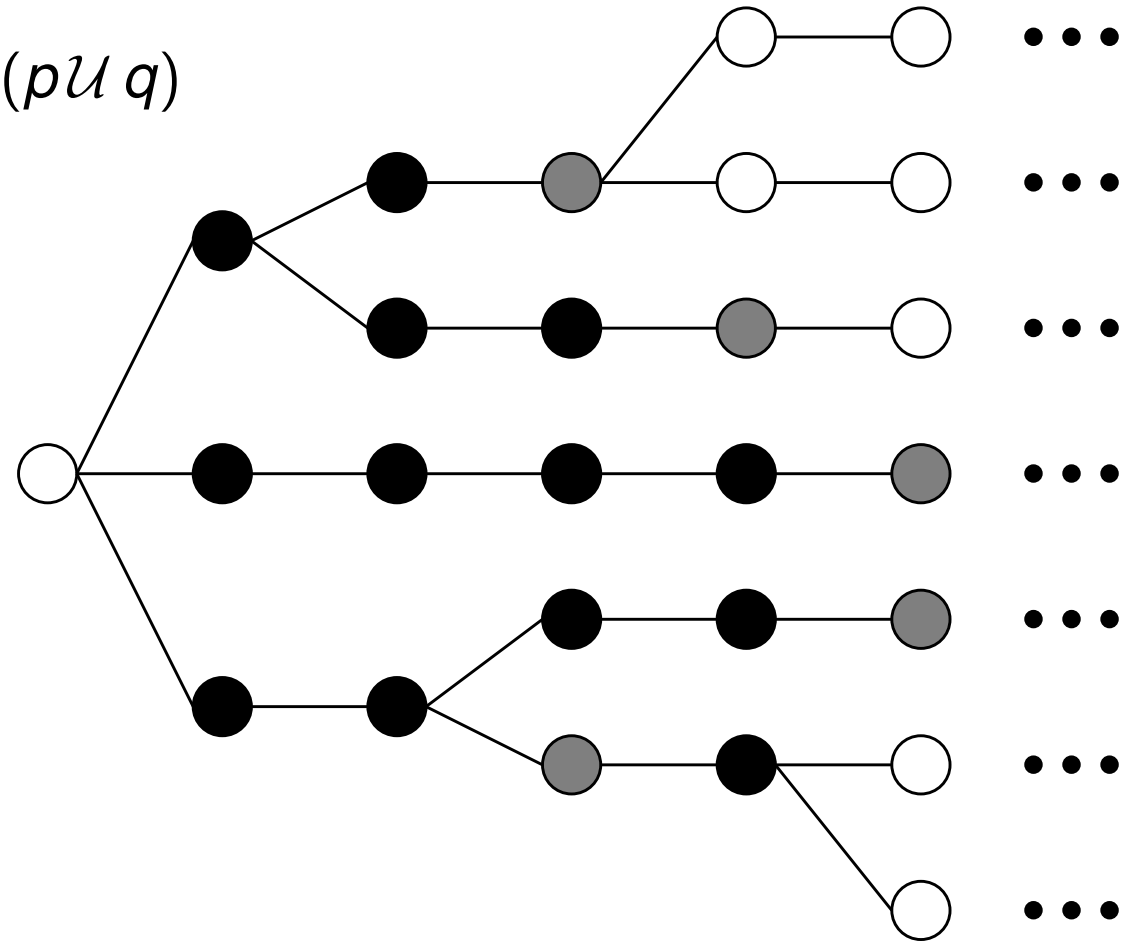
- $s \models A \bigcirc (E \Box p)$



Model Checking

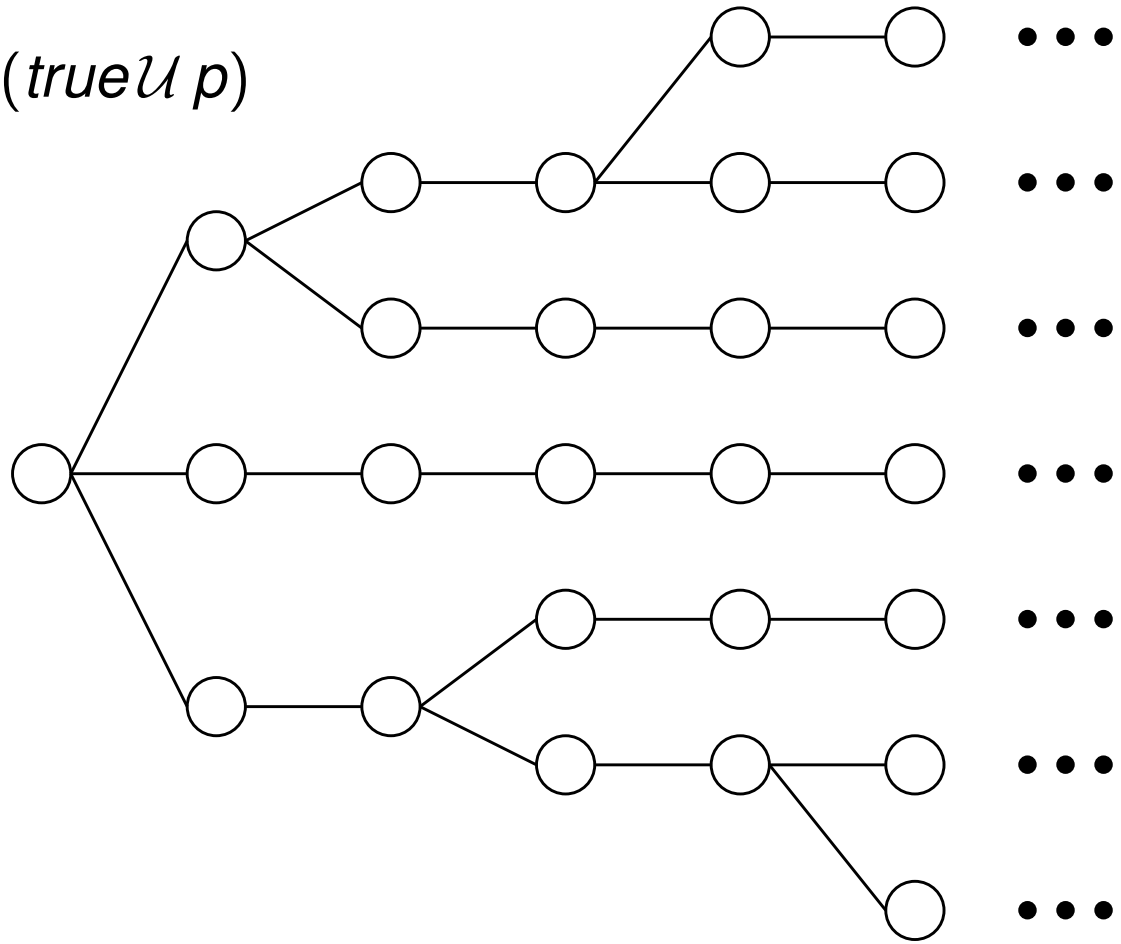
Ejemplos (4/5)

- $s \models A(p \mathcal{U} q)$



Model Checking
Ejemplos (5/5)

- $s \not\models E(true\,\mathcal{U}\,p)$



Model Checking Abstracto

- Vamos a definirlo para CTL*
- Distintas formas de hacerlo: seguiremos *Dams et al.*
- Transiciones de estados mixtas
- Abstracción de programas (no de modelos)
- Método de refinamiento: preservación de resultados

Model Checking Abstracto
La lógica

Añadimos a la lógica CTL* introducida antes, el operador

$$(\psi_1 \mathcal{V} \psi_2) \equiv \neg(\neg\psi_1 \mathcal{U} \neg\psi_2)$$

ψ_2 será cierta mientras ψ_1 sea falsa, y sólo después de que ψ_1 tome el valor cierto, ψ_2 podrá tomar el valor falso

Model Checking Abstracto
La lógica

Añadimos a la lógica CTL* introducida antes, el operador

$$(\psi_1 \mathcal{V} \psi_2) \equiv \neg(\neg\psi_1 \mathcal{U} \neg\psi_2)$$

ψ_2 será cierta mientras ψ_1 sea falsa, y sólo después de que ψ_1 tome el valor cierto, ψ_2 podrá tomar el valor falso

Necesitamos este operador porque sólo permitimos la negación de proposiciones atómicas

Model Checking Abstracto
La lógica

Literales

Una fórmula está basada en proposiciones atómicas *Prop*
Llamamos literales al conjunto de expresiones

$$Lit = Prop \cup \{\neg p | p \in Prop\}$$

Model Checking Abstracto
La lógica

Literales

Una fórmula está basada en proposiciones atómicas *Prop*
Llamamos literales al conjunto de expresiones

$$Lit = Prop \cup \{\neg p | p \in Prop\}$$

Sintaxis, revisión

A partir de los literales podemos definir las *state*-fórmulas:

$$\phi ::= p | \phi \wedge \phi | \phi \vee \phi | A\psi | E\psi$$

y el conjunto de *path*-fórmulas:

$$\psi ::= \phi | \psi \wedge \psi | \psi \vee \psi | \bigcirc \psi | \psi \mathcal{U} \psi | \psi \mathcal{V} \psi$$

Model Checking Abstracto

Simplificando la notación

- $\neg\psi$ equivale a la forma normal negada de la fórmula ψ
- $\mathit{true} \equiv \psi_1 \vee \neg\psi_1$
- $\mathit{false} \equiv \psi_1 \wedge \neg\psi_1$
- $\psi_1 \rightarrow \psi_2 \equiv \neg\psi_1 \vee \psi_2$
- $\Diamond\psi \equiv \mathit{true}\,\mathcal{U}\,\psi$
- $\Box\psi \equiv \mathit{false}\,\mathcal{V}\,\psi \equiv \neg(\mathit{true}\,\mathcal{U}\,\neg\psi)$

Model Checking Abstracto
La lógica

Usaremos dos fragmentos de la lógica CTL*:

- $\forall$CTL*: fórmulas que únicamente contienen cuantificadores universales (**A**)
- $\exists$CTL*: fórmulas que únicamente contienen cuantificadores existenciales (**E**)

Ojo: La fórmula debe estar en forma normal negada

Model Checking Abstracto
La lógica

Usaremos dos fragmentos de la lógica CTL*:

- $\forall$CTL*: fórmulas que únicamente contienen cuantificadores universales (**A**)
- $\exists$CTL*: fórmulas que únicamente contienen cuantificadores existenciales (**E**)

Ojo: La fórmula debe estar en forma normal negada

Model Checking Abstracto

EJERCICIO 1

¿Pertenece $A\Diamond p \rightarrow A\Box q$ al fragmento $\forall$CTL*?

Model Checking Abstracto

EJERCICIO 1

¿Pertenece $A\Diamond p \rightarrow A\Box q$ al fragmento $\forall$CTL*?

Para obtener la forma normal negada de una fórmula usad:

R1: $\neg(\psi_1 \wedge \psi_2) \rightarrow \neg\psi_1 \vee \neg\psi_2$	**R2:** $\neg(\psi_1 \vee \psi_2) \rightarrow \neg\psi_1 \wedge \neg\psi_2$
R3: $\neg A\psi \rightarrow E\neg\psi$	**R4:** $\neg E\psi \rightarrow A\neg\psi$
R5: $\neg \bigcirc \psi \rightarrow \bigcirc\neg\psi$	
R6: $\neg(\psi_1 \,\mathcal{U}\, \psi_2) \rightarrow \neg\psi_1 \,\mathcal{V}\, \neg\psi_2$	**R7:** $\neg(\psi_1 \,\mathcal{V}\, \psi_2) \rightarrow \neg\psi_1 \,\mathcal{U}\, \neg\psi_2$

Model Checking Abstracto

EJERCICIO 2

Tomando como modelo el ejemplo del microondas mostrado antes, responder a las preguntas siguientes:

1. Escribid *siempre que la puerta del microondas esté abierta y el microondas esté encendido, se producirá un error*
2. ¿El modelo satisface $\Box A(\text{cerrado} \wedge \text{on}) \rightarrow A\Diamond(\text{cerrado} \wedge \text{off})$?
3. ¿El modelo satisface $\Box A((\text{abierto} \wedge \text{off}) \vee (\text{cerrado}) \vee ((\text{abierto} \wedge \text{on}) \rightarrow A \bigcirc \text{error}))$?
4. ¿Se satisfaría la condición anterior si hubiera un *arco reflexivo* en el nodo etiquetado como *abierto, on,* $\neg$ *error*?

Model Checking Abstracto

El modelo

Noción *general* de sistemas de transiciones

- Una estructura de *Kripke* es un tipo especial de sistema de transición
- (S, R) siendo S un conjunto de estados y R una relación binaria entre ellos
- Un camino π es una secuencia infinita $\pi = s_0 s_1 \cdots$ para los que $\forall i \in \mathbb{N}, (s_i, s_{i+1}) \in R$
- $\pi(i)$ es el elemento i-ésimo
- π^n sufijo que empieza por $\pi(n)$

Model Checking Abstracto

Una estructura de *Kripke* es pues un sistema de transiciones con, además:

- un conjunto $I \subseteq S$ de estados iniciales, con el que podemos definir una noción de *alcanzabilidad*
- una interpretación: asocia a cada estado los literales que se satisfacen en él
  - asocia a cada literal, los estados en los que se satisface
- la interpretación no puede establecer que tanto p como $\neg p$ se satisfagan en un mismo estado
  - sí que puede ocurrir que ni p ni $\neg p$ se satisfagan en un estado determinado

Model Checking Abstracto

Una estructura de *Kripke* es pues un sistema de transiciones con, además:

- un conjunto $I \subseteq S$ de estados iniciales, con el que podemos definir una noción de *alcanzabilidad*
- una interpretación: asocia a cada estado los literales que se satisfacen en él
  - asocia a cada literal, los estados en los que se satisface
- la interpretación no puede establecer que tanto p como $\neg p$ se satisfagan en un mismo estado
  - sí que puede ocurrir que ni p ni $\neg p$ se satisfagan en un estado determinado

Model Checking Abstracto

EJEMPLO: MATEMÁTICOS

Protocolo de acceso mutuamente exclusivo

Tenemos dos matemáticos que pueden estar en dos estados distintos: *comiendo* o *pensando*.

Para comer, cada matemático consulta una variable del sistema *n*

- si la variable es *par*, el primer matemático podrá comer
- si es *impar*, el que comerá será el segundo matemático.

Una vez terminado de comer, los matemáticos asignaran un valor nuevo a la variable *n*, cada uno de forma distinta.
Inicialmente, los dos matemáticos están pensando y la variable *n* tiene un valor arbitrario.

Model Checking Abstracto

EJEMPLO: MATEMÁTICOS

Protocolo de acceso mutuamente exclusivo

Tenemos dos matemáticos que pueden estar en dos estados distintos: *comiendo* o *pensando*.
Para comer, cada matemático consulta una variable del sistema *n*

- si la variable es *par*, el primer matemático podrá comer
- si es *impar*, el que comerá será el segundo matemático.

Una vez terminado de comer, los matemáticos asignaran un valor nuevo a la variable *n*, cada uno de forma distinta.
Inicialmente, los dos matemáticos están pensando y la variable *n* tiene un valor arbitrario.

Model Checking Abstracto

EJEMPLO: MATEMÁTICOS

Protocolo de acceso mutuamente exclusivo

Tenemos dos matemáticos que pueden estar en dos estados distintos: *comiendo* o *pensando*.
Para comer, cada matemático consulta una variable del sistema *n*

- si la variable es *par*, el primer matemático podrá comer
- si es *impar*, el que comerá será el segundo matemático.

Una vez terminado de comer, los matemáticos asignaran un valor nuevo a la variable *n*, cada uno de forma distinta.
Inicialmente, los dos matemáticos están pensando y la variable *n* tiene un valor arbitrario.

Model Checking Abstracto

EJEMPLO: MATEMÁTICOS

Protocolo de acceso mutuamente exclusivo

Tenemos dos matemáticos que pueden estar en dos estados distintos: *comiendo* o *pensando*.
Para comer, cada matemático consulta una variable del sistema n

- si la variable es *par*, el primer matemático podrá comer
- si es *impar*, el que comerá será el segundo matemático.

Una vez terminado de comer, los matemáticos asignaran un valor nuevo a la variable n, cada uno de forma distinta.
Inicialmente, los dos matemáticos están pensando y la variable n tiene un valor arbitrario.

Model Checking Abstracto

EJEMPLO: MATEMÁTICOS

El objetivo final es el de verificar si el sistema satisface las siguientes propiedades:

- los matemáticos tienen acceso mutuamente excluyente al comedor
- los matemáticos no se bloquean, es decir, si uno está comiendo, tarde o temprano el otro matemático también comerá

Model Checking Abstracto

EJEMPLO: CODIFICACIÓN

- m_0 y m_1 representan el estado en el que se encuentra el primer y segundo matemático respectivamente
- el dominio de m_0 y m_1 es {*pensando*,*comiendo*}
- n es la variable de control y tomará valores en $\mathbb{N}$
- un estado del sistema será una configuración

$$\langle m_0, m_1, n \rangle \in \{pensando, comiendo\}^2 \times \mathbb{N} \backslash \{0\}$$

Model Checking Abstracto

EJEMPLO: CODIFICACIÓN

- m_0 y m_1 representan el estado en el que se encuentra el primer y segundo matemático respectivamente
- el dominio de m_0 y m_1 es {*pensando*,*comiendo*}
- n es la variable de control y tomará valores en $\mathbb{N}$
- un estado del sistema será una configuración

$$\langle m_0, m_1, n\rangle \in \{pensando, comiendo\}^2 \times \mathbb{N}\backslash\{0\}$$

Model Checking Abstracto

EJERCICIO 3

Si tenemos un lenguaje de programación donde definimos acciones que podrán ser tomadas (siempre que se cumpla su condición) de forma no determinista.
El ejemplo de los matemáticos se definiría como:

$$m_0 = \mathit{pensando}, \mathit{odd}(n) \rightarrow m_0 := \mathit{comiendo}$$
$$m_0 = \mathit{comiendo} \rightarrow m_0 := \mathit{pensando}, n := 3 * n + 1$$
$$m_1 = \mathit{pensando}, \mathit{even}(n) \rightarrow m_1 := \mathit{comiendo}$$
$$m_1 = \mathit{comiendo} \rightarrow m_1 := \mathit{pensando}, n := n/2$$

Dibujar el sistema de transiciones asociado, indicando las guardas sobre los arcos.

Model Checking Abstracto

EJERCICIO: SISTEMA DE TRANSICIONES

A completar...

Model Checking Abstracto

EJERCICIO 4

Escribir en CTL* las propiedades que queremos verificar en el ejemplo

Model Checking Abstracto

Se debe garantizar que, al abstraer un modelo, algunas propiedades se preservan de forma que los resultados obtenidos en el mundo abstracto son válidos también en el concreto.

- El modelo abstracto será una versión abstracta de la estructura de *Kripke*
- Si KS es el conjunto de modelos concretos y KS_α el de abstractos, habrá una relación $\xi : KS \times KS_\alpha$ entre ellos
- ξ debe asegurar, al menos, preservación *débil*

Model Checking Abstracto

Se debe garantizar que, al abstraer un modelo, algunas propiedades se preservan de forma que los resultados obtenidos en el mundo abstracto son válidos también en el concreto.

- El modelo abstracto será una versión abstracta de la estructura de *Kripke*
- Si KS es el conjunto de modelos concretos y KS_α el de abstractos, habrá una relación $\xi : KS \times KS_\alpha$ entre ellos
- ξ debe asegurar, al menos, preservación *débil*

Model Checking Abstracto

Preservación débil

La preservación débil dice que la propiedad que pueda ser probada cierta en el abstracto, será cierta en el concreto:

$$\forall K \in KS, A \in KS_\alpha, (\xi(K, A) \Rightarrow \forall \phi \in CTL^*(K \models \phi \Leftarrow A \models \phi))$$

Model Checking Abstracto

Intuición:

- se define una relación ρ entre estados de las estructuras (concreta y abstracta)
- la semántica concreta suele darse en términos de transiciones entre estados
  - definiremos las transiciones de forma que *simulen* el comportamiento real del sistema

Model Checking Abstracto

Preservación débil

Formulación alternativa de la preservación débil:

$$\forall c \in S, a \in S_\alpha, (\rho(c,a) \Rightarrow \forall \phi \in CTL^*((K,c) \models \phi \Leftarrow (A,a) \models \phi))$$

- S conjunto de estados concretos
- S_α conjunto de estados abstractos
- Estamos dentro de un marco con una conexión de Galois
  - asumimos que S_α es un retículo completo con orden $\sqsubseteq$
  - tenemos inserción de Galois de $(\wp(S), \subseteq)$ a $(S_\alpha, \sqsubseteq)$
  - recordemos que $a \sqsubseteq b$ si y sólo si $\gamma(a) \subseteq \gamma(b)$

Model Checking Abstracto

Preservación débil

Formulación alternativa de la preservación débil:

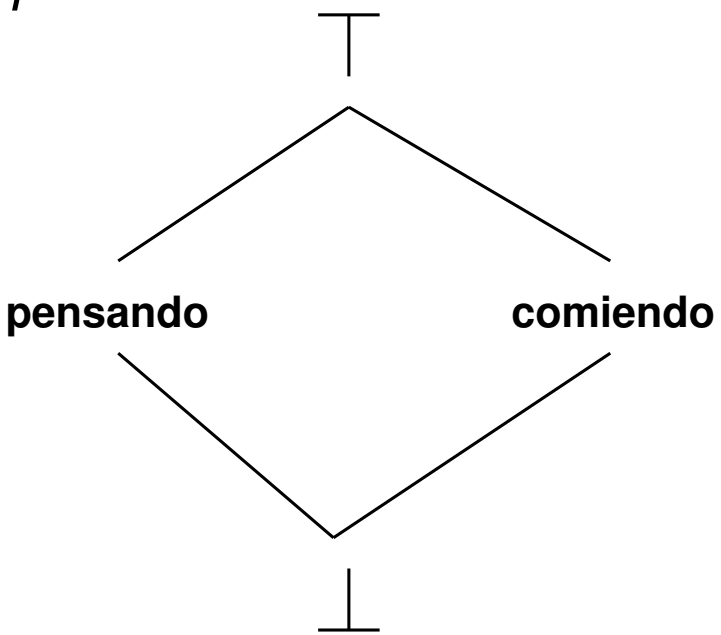
$$\forall c \in S, a \in S_\alpha, (\rho(c,a) \Rightarrow \forall \phi \in CTL^*((K,c) \models \phi \Leftarrow (A,a) \models \phi))$$

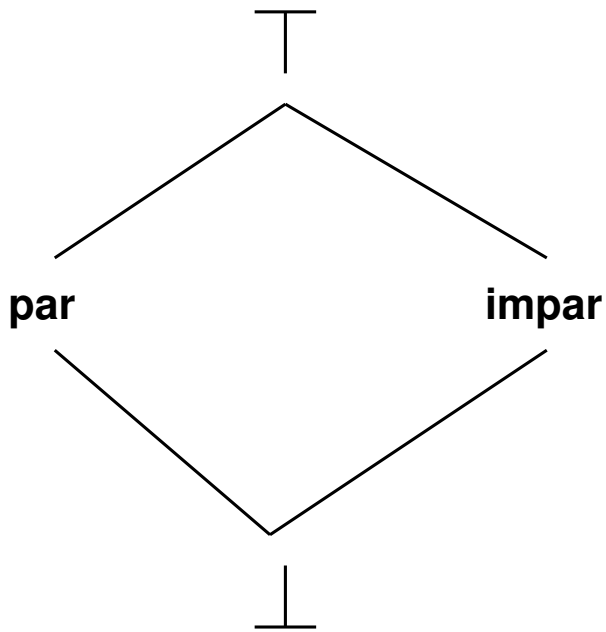
- S conjunto de estados concretos
- S_α conjunto de estados abstractos
- Estamos dentro de un marco con una conexión de Galois
  - asumimos que S_α es un retículo completo con orden $\sqsubseteq$
  - tenemos inserción de Galois de $(\wp(S), \subseteq)$ a $(S_\alpha, \sqsubseteq)$
  - recordemos que $a \sqsubseteq b$ si y sólo si $\gamma(a) \subseteq \gamma(b)$

Model Checking Abstracto

Ejemplo: Dominio abstracto

Vamos a abstraer el sistema de los matemáticos abstrayendo la variable *n*, de forma que describirá los elementos de $\mathbb{N}$ como *par* o *impar*.





Model Checking Abstracto

Ejemplo: Conexión de Galois

$$\alpha(n) \begin{cases} \mathit{par} & \text{si } n \bmod 2 = 0 \\ \mathit{impar} & \text{si } n \bmod 2 \neq 0 \end{cases}$$

$$\gamma(\mathit{par}) = \{2, 4, 6, 8, \ldots\}$$
$$\gamma(\mathit{impar}) = \{1, 3, 5, 7, \ldots\}$$

Ahora tenemos un dominio finito:

$$\Sigma_\alpha = \{\mathit{pensando}, \mathit{comiendo}, \top\}^2 \times \{\mathit{par}, \mathit{impar}, \top\}$$

Model Checking Abstracto

Ejemplo: Conexión de Galois

$$\alpha(n) \begin{cases} \mathit{par} & \text{si } n \bmod 2 = 0 \\ \mathit{impar} & \text{si } n \bmod 2 \neq 0 \end{cases}$$

$$\gamma(\mathit{par}) = \{2, 4, 6, 8, \ldots\}$$
$$\gamma(\mathit{impar}) = \{1, 3, 5, 7, \ldots\}$$

Ahora tenemos un dominio finito:

$$\Sigma_\alpha = \{\mathit{pensando}, \mathit{comiendo}, \top\}^2 \times \{\mathit{par}, \mathit{impar}, \top\}$$

Model Checking Abstracto

Preservación fuerte

La preservación débil conserva las propiedades de seguridad universales.

- si dos estructuras son *isomorfas*, tanto la satisfacción como la no satisfacción de una propiedad será preservada: preservación fuerte
- una relación de bisimulación genera estructuras isomorfas

Model Checking Abstracto

Preservación fuerte

La preservación débil conserva las propiedades de seguridad universales.

- si dos estructuras son *isomorfas*, tanto la satisfacción como la no satisfacción de una propiedad será preservada: preservación fuerte
- una relación de bisimulación genera estructuras isomorfas

Dos definiciones de preservación fuerte:

$$\forall c \in C, a \in A\ (\alpha(c) = a \Rightarrow \forall \phi \in L(c \models \phi \Leftrightarrow a \models^{\alpha} \phi))$$

$$\forall c \in C, a \in A\ (\alpha(c) = a \Leftrightarrow \forall \phi \in L(c \models \phi \Leftrightarrow a \models^{\alpha} \phi))$$

Model Checking Abstracto

Preservación fuerte

Cuando consideremos preservación fuerte, el nivel del simplificación del modelo (al pasar del concreto al abstracto) estará fuertemente determinado por las propiedades que se puedan comprobar

- Ante un mismo lenguaje de propiedades, no habrá reducción de complejidad

Model Checking Abstracto
Preservación fuerte

Cuando consideremos preservación fuerte, el nivel del simplificación del modelo (al pasar del concreto al abstracto) estará fuertemente determinado por las propiedades que se puedan comprobar

- Ante un mismo lenguaje de propiedades, no habrá reducción de complejidad

Estructura de *Kripke* Abstracta

Supongamos que tenemos definido un conjunto de estados abstractos S_α y una inserción de Galois (α, γ) que relaciona S_α con S.

Para definir una Estructura de Kripke Abstracta necesitaremos definir:

1. Una función de interpretación L_α
2. Un conjunto de estados iniciales I_α
3. Una relación de transición R_α entre estados abstractos

Estructura de *Kripke* Abstracta

Supongamos que tenemos definido un conjunto de estados abstractos S_α y una inserción de Galois (α, γ) que relaciona S_α con S.

Para definir una Estructura de Kripke Abstracta necesitaremos definir:

1. Una función de interpretación L_α
2. Un conjunto de estados iniciales I_α
3. Una relación de transición R_α entre estados abstractos

Estructura de *Kripke* Abstracta

Función de interpretación

Para satisfacer la preservación débil, se debe cumplir que para todo literal p,

$$(A, a) \models p \Rightarrow (K, \gamma(a)) \models p$$

Nos interesa que la interpretación asocie a cada estado, cuantos más literales mejor:

Definition (L_α)

Para $p \in Lit, L_\alpha(p) = \{a \in S_\alpha | \gamma(a) \subseteq L(p)\}$

Estructura de *Kripke* Abstracta

Función de interpretación

Para satisfacer la preservación débil, se debe cumplir que para todo literal p,

$$(A, a) \models p \Rightarrow (K, \gamma(a)) \models p$$

Nos interesa que la interpretación asocie a cada estado, cuantos más literales mejor:

Definition (L_α)

Para $p \in Lit, L_\alpha(p) = \{a \in S_\alpha | \gamma(a) \subseteq L(p)\}$

Estructura de *Kripke* Abstracta

Función de interpretación

A partir de L_α se puede definir $\models_\alpha$ de forma similar a cómo se definió $\models$

$\models_\alpha$ cumple que...

para todo $a \in S_\alpha$ y $p \in Lit$, $(A, a) \models p \Leftrightarrow (K, \gamma(a)) \models p$

Estructura de *Kripke* Abstracta

Función de interpretación

A partir de L_α se puede definir $\models_\alpha$ de forma similar a cómo se definió $\models$

$\models_\alpha$ cumple que...

para todo $a \in S_\alpha$ y $p \in Lit$, $(A, a) \models p \Leftrightarrow (K, \gamma(a)) \models p$

Estructura de *Kripke* Abstracta

Función de interpretación

Si $\gamma(a)$ contiene estados concretos, algunos donde se satisface p y otros donde se satisface $\neg p$, entonces ocurrirá que

$$\text{ni } a \models p, \text{ ni } a \models \neg p$$

$$a \not\models p \quad \not\Rightarrow \quad a \models \neg p$$

Estructura de *Kripke* Abstracta

Función de interpretación

Si $\gamma(a)$ contiene estados concretos, algunos donde se satisface p y otros donde se satisface $\neg p$, entonces ocurrirá que

$$\text{ni } a \models p, \text{ ni } a \models \neg p$$

$$a \not\models p \quad \not\Rightarrow \quad a \models \neg p$$

Estructura de *Kripke* Abstracta

Función de interpretación

- cuanto más precisa sea una descripción, más literales satisfará,
- cuanto más abstracta sea una descripción, menos literales satisfará

Precisión

Sean $a, a' \in S_\alpha$, si $a' \sqsupseteq a$, entonces para todo $p \in Lit$, $a' \models p \Rightarrow a \models p$.

Estructura de *Kripke* Abstracta

Función de interpretación

- cuanto más precisa sea una descripción, más literales satisfará,
- cuanto más abstracta sea una descripción, menos literales satisfará

Precisión

Sean $a, a' \in S_\alpha$, si $a' \sqsupseteq a$, entonces para todo $p \in Lit$, $a' \models p \Rightarrow a \models p$.

Estructura de *Kripke* Abstracta
Estados Iniciales

Idea

Teniendo en cuenta las condiciones de preservación, claramente el conjunto de estados iniciales concretos debe estar representado por los estados iniciales abstractos

- I_α tendrá que ser mínimo para garantizar la eficiencia del método (precisión).
- Situación ideal: $I = \cup\{\gamma(a) | a \in I_\alpha\}$
  - en general imposible de cumplir

Estructura de *Kripke* Abstracta

Estados Iniciales

Idea

Teniendo en cuenta las condiciones de preservación, claramente el conjunto de estados iniciales concretos debe estar representado por los estados iniciales abstractos

- I_α tendrá que ser mínimo para garantizar la eficiencia del método (precisión).
- Situación ideal: $I = \cup\{\gamma(a)|a \in I_\alpha\}$
  - en general imposible de cumplir

Estructura de *Kripke* Abstracta

Estados Iniciales

Definición del conjunto mínimo de estados iniciales:

$$I_\alpha = \{\alpha(c) | c \in I\}$$

Definición no equivalente: $I'_\alpha = \alpha(I)$

- I'_α es, en general *menos precisa* que I_α
- ver ejemplo siguiente...

Estructura de *Kripke* Abstracta

Estados Iniciales

Definición del conjunto mínimo de estados iniciales:

$$I_\alpha = \{\alpha(c) | c \in I\}$$

Definición no equivalente: $I'_\alpha = \alpha(I)$

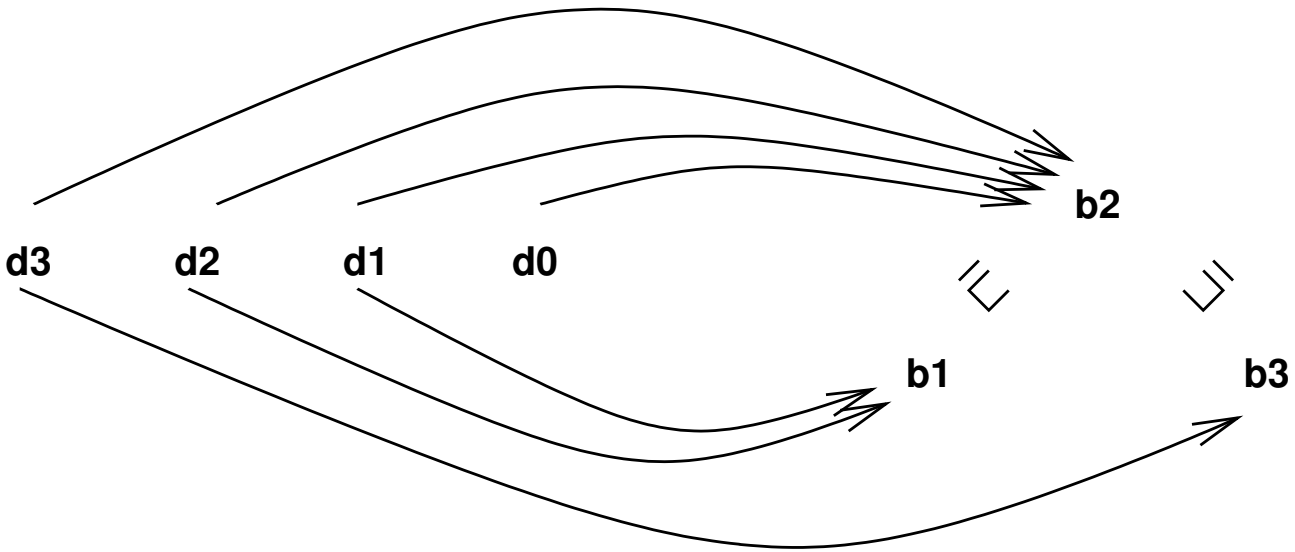
- I'_α es, en general *menos precisa* que I_α
- ver ejemplo siguiente...

Estructura de *Kripke* Abstracta

Estados Iniciales

EJEMPLO

Supongamos que en la siguiente figura, $I = \{d_3, d_2, d_1\}$:

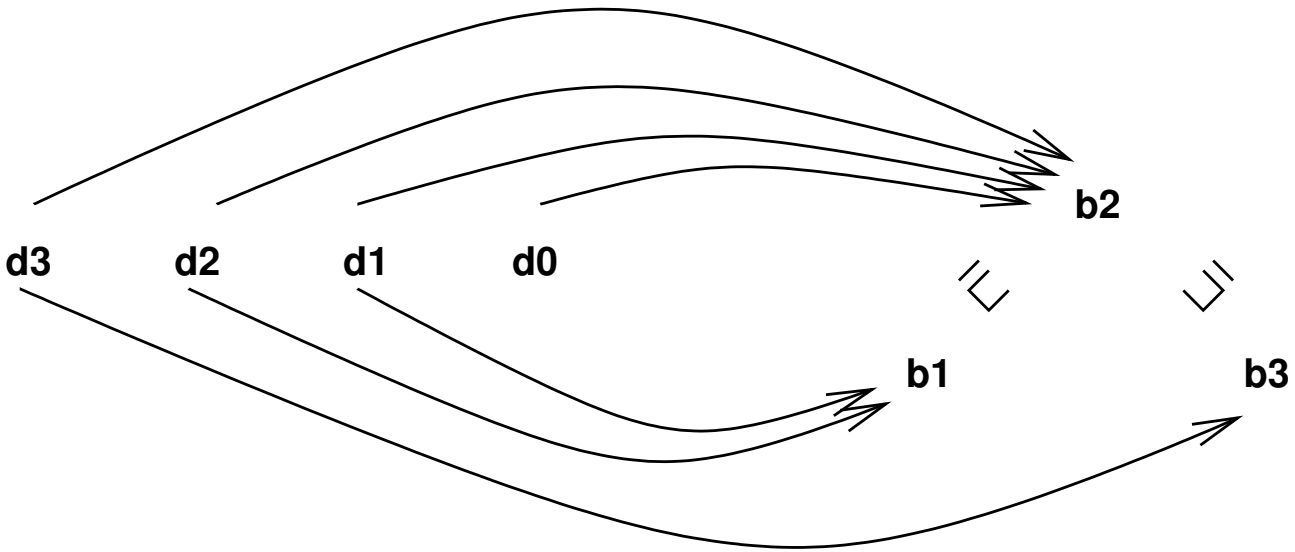


Estructura de *Kripke* Abstracta

Estados Iniciales

EJEMPLO

Supongamos que en la siguiente figura, $I = \{d_3, d_2, d_1\}$:



- $I_\alpha = \{b_1, b_3\}$, por lo que $\gamma(I_\alpha) = \{d_3, d_2, d_1\}$
- $I'_\alpha = \{b_2\}$, por lo que $\gamma(I'_\alpha) = \{d_3, d_2, d_1, d_0\}$

Estructura de *Kripke* Abstracta

Relación de transición

¿Cuándo debemos definir una transición entre dos estados en nuestro modelo abstracto?

- para preservar propiedades existenciales: si existe una transición de a que cumpla una propiedad, implicará que $\forall c \in \gamma(a)$, existe un sucesor de c en el modelo concreto que satisface la propiedad considerada
- para preservar propiedades universales: el que todos los sucesores de a satisfagan una propiedad, implica que dicha propiedad se cumple para todo sucesor de $\forall c \in \gamma(a)$

Estructura de *Kripke* Abstracta

Relación de transición

¿Cuándo debemos definir una transición entre dos estados en nuestro modelo abstracto?

- para preservar propiedades existenciales: si existe una transición de a que cumpla una propiedad, implicará que $\forall c \in \gamma(a)$, existe un sucesor de c en el modelo concreto que satisface la propiedad considerada
- para preservar propiedades universales: el que todos los sucesores de a satisfagan una propiedad, implica que dicha propiedad se cumple para todo sucesor de $\forall c \in \gamma(a)$

Estructura de *Kripke* Abstracta

Relación de transición

¿Cuándo debemos definir una transición entre dos estados en nuestro modelo abstracto?

- para preservar propiedades existenciales: si existe una transición de a que cumpla una propiedad, implicará que $\forall c \in \gamma(a)$, existe un sucesor de c en el modelo concreto que satisface la propiedad considerada
- para preservar propiedades universales: el que todos los sucesores de a satisfagan una propiedad, implica que dicha propiedad se cumple para todo sucesor de $\forall c \in \gamma(a)$

Estructura de *Kripke* Abstracta

Relación de transición

Una relación que satisfaga las dos condiciones anteriores es una bisimulación

- requisito demasiado costoso
- penaliza la *simplificación* del modelo
- implicaría que toda fórmula CTL* se preservaría

Estructura de *Kripke* Abstracta

Relación de transición

Una relación que satisfaga las dos condiciones anteriores es una bisimulación

- requisito demasiado costoso
- penaliza la *simplificación* del modelo
- implicaría que toda fórmula CTL* se preservaría

Estructura de *Kripke* Abstracta

Relación de transición

SOLUCIÓN

Se definen dos relaciones en lugar de una

- una preservará las propiedades existenciales
  - podremos analizar sólo esta relación cuando estemos comprobando propiedades de $\exists$CTL*
- la otra preservará propiedades universales
  - podremos analizar sólo esta relación cuando estemos comprobando propiedades de $\forall$CTL*

Estructura de *Kripke* Abstracta

Relación de transición

SOLUCIÓN

Se definen dos relaciones en lugar de una

- una preservará las propiedades existenciales
  - podremos analizar sólo esta relación cuando estemos comprobando propiedades de $\exists$CTL*
- la otra preservará propiedades universales
  - podremos analizar sólo esta relación cuando estemos comprobando propiedades de $\forall$CTL*

Puede darse el caso en el que ni ϕ ni $\neg\phi$ se satisfagan en el modelo abstracto

- no tenemos preservación *total* de CTL*

Estructura de *Kripke* Abstracta

Relación de transición

Las dos relaciones del modelo abstracto se llaman:

- relación de transición vinculada
- relación de transición libre

Estructura de *Kripke* Abstracta

Relación de transición vinculada

DEFINICIÓN AUXILIARES

Definition

Las relaciones $R^{\exists\exists}, R^{\forall\exists} \subseteq \wp(A) \times \wp(B)$ se definen como:

- $R^{\exists\exists} = \{(X, Y) | \exists x \in X \exists y \in Y . R(X, Y)\}$
- $R^{\forall\exists} = \{(X, Y) | \forall x \in X \exists y \in Y . R(X, Y)\}$

Estructura de *Kripke* Abstracta

Relación de transición vinculada

Dados

- $a \in S_\alpha$
- $b \in S_\alpha$ sucesor de a que satisface ϕ
- $a \models \exists \bigcirc \phi$

Para garantizar la preservación: todo estado concreto $c \in \gamma(a)$ deberá tener un sucesor d que satisfaga ϕ

Estructura de *Kripke* Abstracta

Relación de transición vinculada

Dados

- $a \in S_\alpha$
- $b \in S_\alpha$ sucesor de a que satisface ϕ
- $a \models \exists \bigcirc \phi$

Para garantizar la preservación: todo estado concreto $c \in \gamma(a)$ deberá tener un sucesor d que satisfaga ϕ

b será sucesor de a sólo si $R^{\forall\exists}(\gamma(a), Y)$ para un conjunto $Y \subseteq S$ cuya descripción es b $(\gamma(b) \supseteq Y)$

Estructura de *Kripke* Abstracta

Relación de transición vinculada

- La relación anterior es segura: conserva las propiedades existenciales
  pero...
- nos interesa que *a* tenga el máximo número de sucesores
  - debemos definir los sucesores como las descripciones de *Y* más precisas posibles

Estructura de *Kripke* Abstracta

Relación de transición vinculada

- La relación anterior es segura: conserva las propiedades existenciales
  pero...
- nos interesa que *a* tenga el máximo número de sucesores
  - debemos definir los sucesores como las descripciones de *Y* más precisas posibles

Estructura de *Kripke* Abstracta

Relación de transición vinculada

- La relación anterior es segura: conserva las propiedades existenciales
  pero...
- nos interesa que *a* tenga el máximo número de sucesores
  - debemos definir los sucesores como las descripciones de *Y* más precisas posibles

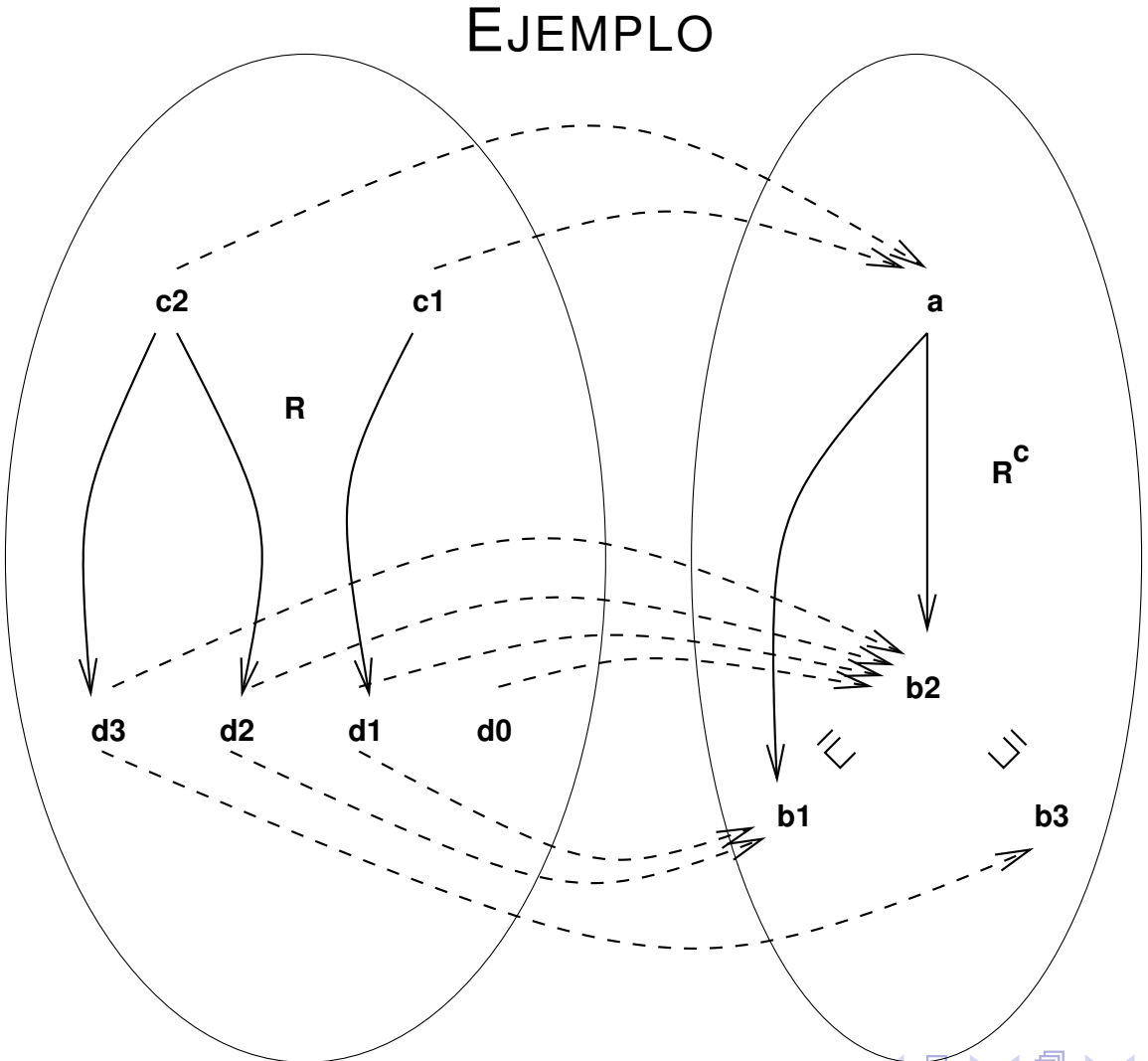
Definition

La relación de transición abstracta vinculada R_α^C se define como

$$R_\alpha^C(a,b) \Leftrightarrow b \in \{\alpha(Y) | Y \in \min\{Y' | R^{\forall\exists}(\gamma(a), Y')\}\}$$

Estructura de *Kripke* Abstracta

Relación de transición vinculada



Estructura de *Kripke* Abstracta

Relación de transición vinculada

Se cumple que

- si $a \in S_\alpha$
- si $c \in \gamma(a)$
- si s_α es un camino en la estructura abstracta sobre la relación vinculada R_α^C

entonces existe un camino s en $\gamma(s_\alpha)$ que se corresponde con una traza en el modelo concreto

Estructura de *Kripke* Abstracta

Relación de transición vinculada

Se cumple que

- si $a \in S_\alpha$
- si $c \in \gamma(a)$
- si s_α es un camino en la estructura abstracta sobre la relación vinculada R_α^C

entonces existe un camino s en $\gamma(s_\alpha)$ que se corresponde con una traza en el modelo concreto

Estructura de *Kripke* Abstracta

Relación de transición libre

Dados

- $a \in S_\alpha$
- todo $b \in S_\alpha$ sucesor de a satisface ϕ
- $a \models \forall \bigcirc \phi$

Para garantizar la preservación: todo sucesor de todo estado concreto $c \in \gamma(a)$ deberá satisfacer ϕ

Estructura de *Kripke* Abstracta

Relación de transición libre

Dados

- $a \in S_\alpha$
- todo $b \in S_\alpha$ sucesor de a satisface ϕ
- $a \models \forall \bigcirc \phi$

Para garantizar la preservación: todo sucesor de todo estado concreto $c \in \gamma(a)$ deberá satisfacer ϕ

b será sucesor de a sólo si $R^{\exists\exists}(\gamma(a), Y)$ para un conjunto $Y \subseteq S$ cuya descripción es b ($\gamma(b) \supseteq Y$)

Estructura de *Kripke* Abstracta

Relación de transición libre

- La relación anterior es segura: conserva las propiedades universales
  pero...
- nos interesa que *a* tenga el mínimo número de sucesores
  - debemos definir los sucesores como las descripciones de *Y* más precisas posibles

Estructura de *Kripke* Abstracta

Relación de transición libre

- La relación anterior es segura: conserva las propiedades universales
  pero...
- nos interesa que *a* tenga el mínimo número de sucesores
  - debemos definir los sucesores como las descripciones de *Y* más precisas posibles

Estructura de *Kripke* Abstracta

Relación de transición libre

- La relación anterior es segura: conserva las propiedades universales
  pero...
- nos interesa que *a* tenga el mínimo número de sucesores
  - debemos definir los sucesores como las descripciones de Y más precisas posibles

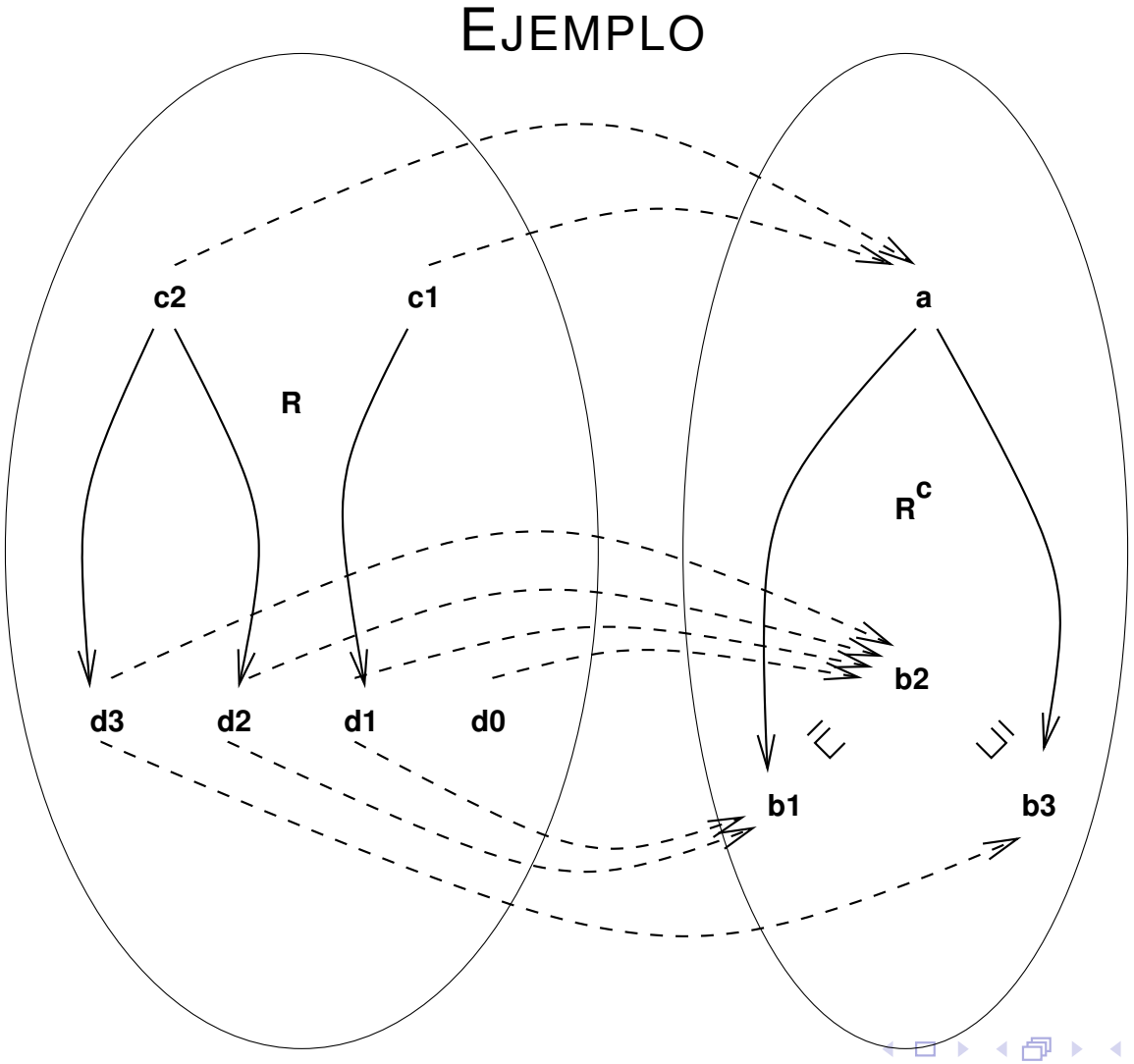
Definition

La relación de transición abstracta libre R_α^F se define como

$$R_\alpha^F(a,b) \Leftrightarrow b \in \{\alpha(Y) | Y \in \min\{Y' | R^{\exists\exists}(\gamma(a), Y')\}\}$$

Estructura de *Kripke* Abstracta

Relación de transición libre



Estructura de *Kripke* Abstracta

Relación de transición libre

Se cumple que

- si $a \in S_\alpha$
- si $c \in \gamma(a)$
- si s es un camino en el modelo concreto

entonces existe un camino s_α sobre la relación R_α^F que describe s

Estructura de *Kripke* Abstracta

Relación de transición libre

Se cumple que

- si $a \in S_\alpha$
- si $c \in \gamma(a)$
- si s es un camino en el modelo concreto

entonces existe un camino s_α sobre la relación R_α^F que describe s

Estructura de *Kripke* Abstracta

Relación de transición libre

Debido a la condición impuesta tanto a la relación vinculada como a la libre de minimalidad del conjunto Y, en general no se cumple que $R^C\alpha \subseteq R^F_\alpha$

Estructura de *Kripke* Abstracta

Sistema de transición mixto

Concepto que integra las dos relaciones definidas:

Sistema de transición mixto

Tripla $\langle S, F, C \rangle$ donde

- S conjunto de estados
- C relación de transición vinculada
- F relación de transición libre

Caminos:

- camino libre: transiciones sobre F
- camino vinculado: transiciones sobre C

Estructura de *Kripke* Abstracta

Sistema de transición mixto

Concepto que integra las dos relaciones definidas:

Sistema de transición mixto

Tripla $\langle S, F, C \rangle$ donde

- S conjunto de estados
- C relación de transición vinculada
- F relación de transición libre

Caminos:

- camino libre: transiciones sobre F
- camino vinculado: transiciones sobre C

Estructura de *Kripke* Abstracta

Sistema de transición mixta

Interpretación de fórmulas CTL* sobre sistemas de transición mixtos:

- $s \models \forall\psi$ si y sólo si para todo camino libre π, $\pi \models \psi$
- $s \models \exists\psi$ si y sólo si existe un camino vinculado π tal que $\pi \models \psi$

Estructura de *Kripke* Abstracta

Estructura de Kripke Abstracta

Es una quintupla $(S_\alpha, F, C, J, L_\alpha)$ representando un sistema de transiciones mixto con

- J conjunto de estados iniciales
- L función de interpretación
- con función de alcanzabilidad definida incluyendo $F \cup C$

Estructura de *Kripke* Abstracta

$$\alpha^M : KS \rightarrow KS_\alpha$$

mapea la estructura K a la estructura abstracta

$$K_\alpha = (S_\alpha, R^F_\alpha, R^C_\alpha, I_\alpha, L_\alpha)$$

donde R^C_α, R^F_α, I_α y L_α se definen como hemos descrito antes

Estructura de *Kripke* Abstracta

Teorema

Para toda fórmula $\phi \in CTL^*$, $\alpha^M(K) \models \phi \Rightarrow K \models \phi$

Abstracción de programas

IDEA

- no queremos construir el modelo concreto para luego pasar al abstracto
- la semántica de un programa suele darse mediante sistema de transiciones entre estados
- queremos dar una semántica alternativa (no estándar) que genere el modelo abstracto directamente
  - la traza del programa ahora será una traza abstracta

Abstracción de programas

La definición de la semántica abstracta dependerá completamente del lenguaje de programación en cuestión.

- Podemos definir la semántica de un lenguaje calculando sus puntos fijos y abstraerla
- Podemos definir puntos fijos abstractos que definan la semántica abstracta del lenguaje

Abstracción de programas

- Sea $(C, \subseteq)$ un cpo
- $f : C \to C$ una función monótona (definiendo la semántica del lenguaje)
- $(A, \leq)$ un poset
- $f_\alpha : A \to A$ monótona
- Sea (α, γ) una inserción de Galois

Si

$$\alpha \circ f \leq f_\alpha \circ \alpha$$

entonces

$$\alpha(\mathrm{lfp}(f)) \leq \mathrm{lfp}(f_\alpha)$$

Abstracción de programas

- Sea $(C, \subseteq)$ un cpo
- $f : C \to C$ una función monótona (definiendo la semántica del lenguaje)
- $(A, \leq)$ un poset
- $f_\alpha : A \to A$ monótona
- Sea (α, γ) una inserción de Galois

Si

$$\alpha \circ f \leq f_\alpha \circ \alpha$$

entonces

$$\alpha(\mathrm{lfp}(f)) \leq \mathrm{lfp}(f_\alpha)$$

Abstracción de programas

Nos centramos en un lenguaje de programación concreto: lenguaje de acciones con guardas

- un programa es un conjunto de acciones de la forma $c_i(\overline{x}) \rightarrow t_i(\overline{x}, \overline{x}')$,
  - i es un valor de un conjunto indexado J
  - $\overline{x}$ es un vector de variables del sistema
  - c_i es una condición
  - t_i representa la actualización de valores de las variables

El programa se ejecuta eligiendo de forma no determinista una acción para la que se cumple su condición

Abstracción de programas

Nos centramos en un lenguaje de programación concreto: lenguaje de acciones con guardas

- un programa es un conjunto de acciones de la forma $c_i(\overline{x}) \rightarrow t_i(\overline{x}, \overline{x}')$,
  - i es un valor de un conjunto indexado J
  - $\overline{x}$ es un vector de variables del sistema
  - c_i es una condición
  - t_i representa la actualización de valores de las variables

El programa se ejecuta eligiendo de forma no determinista una acción para la que se cumple su condición

Abstracción de programas

- Val: Dominio de las variables del sistema
- IVal: Valores iniciales de las variables del sistema
- $\mathrm{IVal} \subseteq \mathrm{Val}$
- c_i es un predicado sobre Val
- t_i es una relación sobre $\mathrm{Val} \times \mathrm{Val}$

Abstracción de programas

Semántica estándar

La función de interpretación concreta $\mathcal{I} : Lang \rightarrow KS$ se define como sigue. Sea $P = \{c_i(\overline{x}) \rightarrow t_i(\overline{x}, \overline{x}')|i \in J\}$ en Lang, $I(P)$ es el sistema de transición (S, I, R) donde:

- S= Val
- I= IVal
- $R = \{(\overline{v}, \overline{v}') \in \mathrm{Val}^2 | \exists i \in J.c_i(\overline{v}) \wedge t_i(\overline{v}, \overline{v}')\}$.

Abstracción de programas

Tenemos que definir la semántica abstracta bajo un marco de inserción de Galois (α, γ) de $(\wp(\mathrm{Val}, \subseteq)$ a $(\mathrm{Val}_\alpha, \sqsubseteq)$

Hay que definir la semántica libre y vinculada para cada predicado del programa

Abstracción de programas

Tenemos que definir la semántica abstracta bajo un marco de inserción de Galois (α, γ) de $(\wp(\mathrm{Val}, \subseteq)$ a $(\mathrm{Val}_\alpha, \sqsubseteq)$

Hay que definir la semántica libre y vinculada para cada predicado del programa

Abstracción de programas

Para $i \in J$, sea c_i^F, c_i^C las condiciones sobre Val_α y t_i^F y t_i^C las transformaciones sobre $Val_\alpha \times Val_\alpha$:

- $c_i^F(a)$ interpretación libre de c_i sii $\forall a \in \mathrm{Val}_\alpha$,

$$c_i^F(a) \Leftrightarrow \exists \overline{v} \in \gamma(a).c_i(\overline{v})$$

- $t_i^F(a)$ interpretación libre de t_i sii $\forall a, b \in \mathrm{Val}_\alpha$,

$$t_i^F(a,b) \Leftrightarrow b \in \{\alpha(Y) | Y \in \min\{Y' | t_i^{\exists\exists}(\gamma(a), Y')\}\}$$

- $c_i^C(a)$ interpretación vinculada de c_i sii $\forall a \in \mathrm{Val}_\alpha$,

$$c_i^C(a) \Leftrightarrow \forall \overline{v} \in \gamma(a).c_i(\overline{v})$$

- $t_i^C(a)$ interpretación vinculada de t_i sii $\forall a, b \in \mathrm{Val}_\alpha$,

$$t_i^C(a,b) \Leftrightarrow b \in \{\alpha(Y) | Y \in \min\{Y' | t_i^{\forall\exists}(\gamma(a), Y')\}\}$$

Abstracción de programas

Para $i \in J$, sea c_i^F, c_i^C las condiciones sobre Val_α y t_i^F y t_i^C las transformaciones sobre $Val_\alpha \times Val_\alpha$:

- $c_i^F(a)$ interpretación libre de c_i sii $\forall a \in \mathrm{Val}_\alpha$,

$$c_i^F(a) \Leftrightarrow \exists \overline{v} \in \gamma(a).c_i(\overline{v})$$

- $t_i^F(a)$ interpretación libre de t_i sii $\forall a, b \in \mathrm{Val}_\alpha$,

$$t_i^F(a,b) \Leftrightarrow b \in \{\alpha(Y) | Y \in \min\{Y' | t_i^{\exists\exists}(\gamma(a), Y')\}\}$$

- $c_i^C(a)$ interpretación vinculada de c_i sii $\forall a \in \mathrm{Val}_\alpha$,

$$c_i^C(a) \Leftrightarrow \forall \overline{v} \in \gamma(a).c_i(\overline{v})$$

- $t_i^C(a)$ interpretación vinculada de t_i sii $\forall a, b \in \mathrm{Val}_\alpha$,

$$t_i^C(a,b) \Leftrightarrow b \in \{\alpha(Y) | Y \in \min\{Y' | t_i^{\forall\exists}(\gamma(a), Y')\}\}$$

Abstracción de programas

La semántica $\mathcal{I}_\alpha(P)$ de un programa P se define como el sistema $\widehat{A^M} = (S_\alpha, \widehat{R^F_\alpha}, \widehat{R^C_\alpha}, I_\alpha)$ donde:

- $S_\alpha = \mathrm{Val}_\alpha$
- $\widehat{R^F_\alpha} = \{(a,b) \in \mathrm{Val}^2 | \exists i \in J. c_i^F(a) \wedge t_i^F(a,b)\}$.
- $\widehat{R^C_\alpha} = \{(a,b) \in \mathrm{Val}^2 | \exists i \in J. c_i^C(a) \wedge t_i^C(a,b)\}$.
- $I_\alpha = \{\alpha(\overline{v}) | \overline{v} \in \mathrm{IVal}\}$

Abstracción de programas

La semántica $\mathcal{I}_\alpha(P)$ de un programa P se define como el sistema $\widehat{A^M} = (S_\alpha, \widehat{R^F_\alpha}, \widehat{R^C_\alpha}, I_\alpha)$ donde:

- $S_\alpha = \mathrm{Val}_\alpha$
- $\widehat{R^F_\alpha} = \{(a,b) \in \mathrm{Val}^2 | \exists i \in J. c_i^F(a) \wedge t_i^F(a,b)\}$.
- $\widehat{R^C_\alpha} = \{(a,b) \in \mathrm{Val}^2 | \exists i \in J. c_i^C(a) \wedge t_i^C(a,b)\}$.
- $I_\alpha = \{\alpha(\overline{v}) | \overline{v} \in \mathrm{IVal}\}$

$\widehat{R^F_\alpha}$ y $\widehat{R^C_\alpha}$ son las llamadas *relaciones de transiciones computadas libre y vinculada* respectivamente.

Abstracción de programas

Ejemplo

Dominio de estados:

$$\Sigma_\alpha = S_\alpha = \{\mathit{pensando}, \mathit{comiendo}, \top\}^2 \times \{\mathit{par}, \mathit{impar}, \top\}$$

Estados iniciales:

$$I_\alpha = \{\langle \mathit{pensando}, \mathit{pensando}, \mathit{par}\rangle, \langle \mathit{pensando}, \mathit{pensando}, \mathit{impar}\rangle\}$$

Abstracción de programas

Ejemplo

Dominio de estados:

$$\Sigma_\alpha = S_\alpha = \{\mathit{pensando}, \mathit{comiendo}, \top\}^2 \times \{\mathit{par}, \mathit{impar}, \top\}$$

Estados iniciales:

$$I_\alpha = \{\langle \mathit{pensando}, \mathit{pensando}, \mathit{par}\rangle, \langle \mathit{pensando}, \mathit{pensando}, \mathit{impar}\rangle\}$$

Abstracción de programas

EJEMPLO

Dominio de estados:

$$\Sigma_\alpha = S_\alpha = \{\mathit{pensando}, \mathit{comiendo}, \top\}^2 \times \{\mathit{par}, \mathit{impar}, \top\}$$

Estados iniciales:

$$I_\alpha = \{\langle \mathit{pensando}, \mathit{pensando}, \mathit{par}\rangle, \langle \mathit{pensando}, \mathit{pensando}, \mathit{impar}\rangle\}$$

Coincide con la definición $I_\alpha = \{\alpha(c) | c \in I\}$

Abstracción de programas

EJEMPLO: INTERPRETACIÓN LIBRE (1/2)

FREE	(*par*,*par*)	(*par*,*impar*)	(*par*,$\top$)	(*impar*,*par*)
$3*^F$	*true*	*false*	*false*	*false*
$+1^F$	*false*	*true*	*false*	*true*
$/2^F$	*true*	*true*	*false*	*false*

(*impar*,*impar*)	(*impar*,$\top$)	($\top$,*par*)	($\top$,*impar*)	($\top$,$\top$)
true	*false*	*true*	*true*	*false*
false	*false*	*true*	*true*	*false*
false	*false*	*true*	*true*	*false*

Abstracción de programas

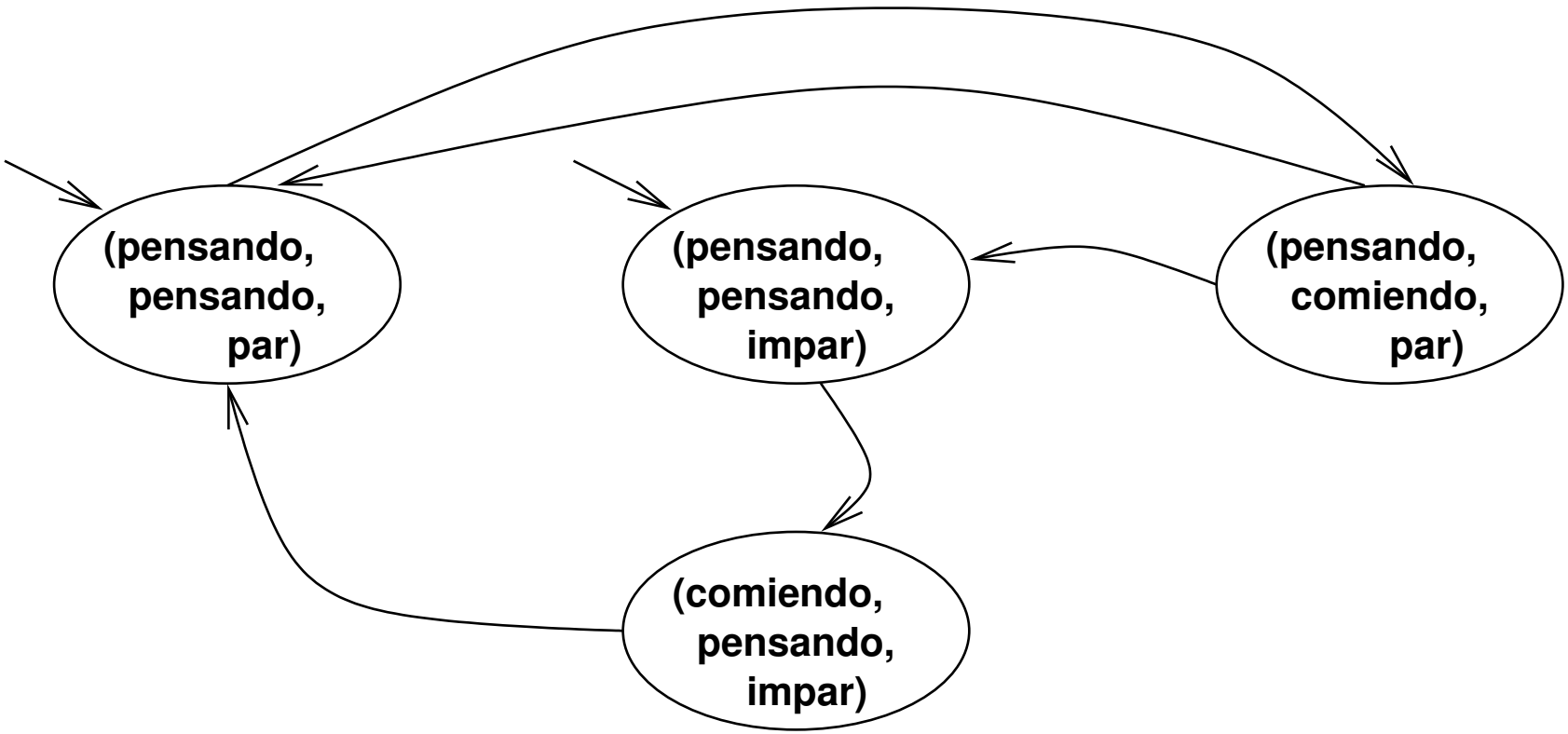
EJEMPLO: INTERPRETACIÓN LIBRE (2/2)

FREE	*par*	*impar*	$\top$
par^F	*true*	*false*	*true*
$impar^F$	*false*	*true*	*true*

FREE	*pensando*	*comiendo*	$\top$
$(=pensando)^F$	*true*	*false*	*true*
$(=comiendo)^F$	*false*	*true*	*true*

Abstracción de programas

EJEMPLO: MODELO ABSTRACTO LIBRE



Abstracción de programas

Ejemplo: El modelo completo (1/5)

Añadimos un proceso que reinicia el protocolo:

$$m_0 = \mathit{pensando}, m_1 = \mathit{pensando} \rightarrow n := 100$$

Queremos comprobar que a lo largo de todo camino, a partir de cualquier estado sale un camino que alcanza un estado de *reinicio*

$$\mathit{reset} \equiv m_0 = \mathit{pensando} \wedge m_1 = \mathit{pensando} \wedge n = 100$$

$$\forall\Box\exists\Diamond \mathit{reset}$$

Abstracción de programas

EJEMPLO: EL MODELO COMPLETO (1/5)

Añadimos un proceso que reinicia el protocolo:

$$m_0 = \mathit{pensando}, m_1 = \mathit{pensando} \rightarrow n := 100$$

Queremos comprobar que a lo largo de todo camino, a partir de cualquier estado sale un camino que alcanza un estado de *reinicio*

$$\mathit{reset} \equiv m_0 = \mathit{pensando} \wedge m_1 = \mathit{pensando} \wedge n = 100$$

$$\forall\Box\exists\Diamond \mathit{reset}$$

Abstracción de programas

Ejemplo: El modelo completo (2/5)

- $\gamma(100) = \{100\}$
- La propiedad es CTL*
- Necesitamos la relación de transición vinculada, no basta la libre
- Se extienden las tablas anteriores con el nuevo valor abstracto 100
- Se definen las tablas para la relación vinculada (mostramos sólo los valores relevantes)

Abstracción de programas

Ejemplo: El modelo completo (2/5)

- $\gamma(100) = \{100\}$
- La propiedad es CTL*
- Necesitamos la relación de transición vinculada, no basta la libre
- Se extienden las tablas anteriores con el nuevo valor abstracto 100
- Se definen las tablas para la relación vinculada (mostramos sólo los valores relevantes)

Abstracción de programas

Ejemplo: El modelo completo (2/5)

- $\gamma(100) = \{100\}$
- La propiedad es CTL*
- Necesitamos la relación de transición vinculada, no basta la libre
- Se extienden las tablas anteriores con el nuevo valor abstracto 100
- Se definen las tablas para la relación vinculada (mostramos sólo los valores relevantes)

Abstracción de programas

EJEMPLO: EL MODELO COMPLETO (3/5)

CONSTR.	*(par,100)*	*(par,par)*	*(par,impar)*	*(par,⊤)*
$/2^C$	*false*	*false*	*false*	*true*

(100,100)	*(100,par)*	*(100,impar)*	*(100,⊤)*
false	*true*	*false*	*false*

CONSTR.	*(impar,100)*	*(impar,par)*	*(impar,impar)*	*(impar,⊤)*
$3*^C$	*false*	*false*	*true*	*false*
$+1^C$	*false*	*true*	*false*	*false*

Abstracción de programas

EJEMPLO: EL MODELO COMPLETO (4/5)

CONSTR.	*100*	*par*	*impar*	$\top$
$even^C$	*true*	*true*	*false*	*false*
odd^C	*false*	*false*	*true*	*false*

CONSTR.	*pensando*	*comiendo*	$\top$
$= pensando^C$	*true*	*false*	*false*
$= comiendo^C$	*false*	*true*	*false*

Abstracción de programas

EJEMPLO: EL MODELO COMPLETO (5/5)