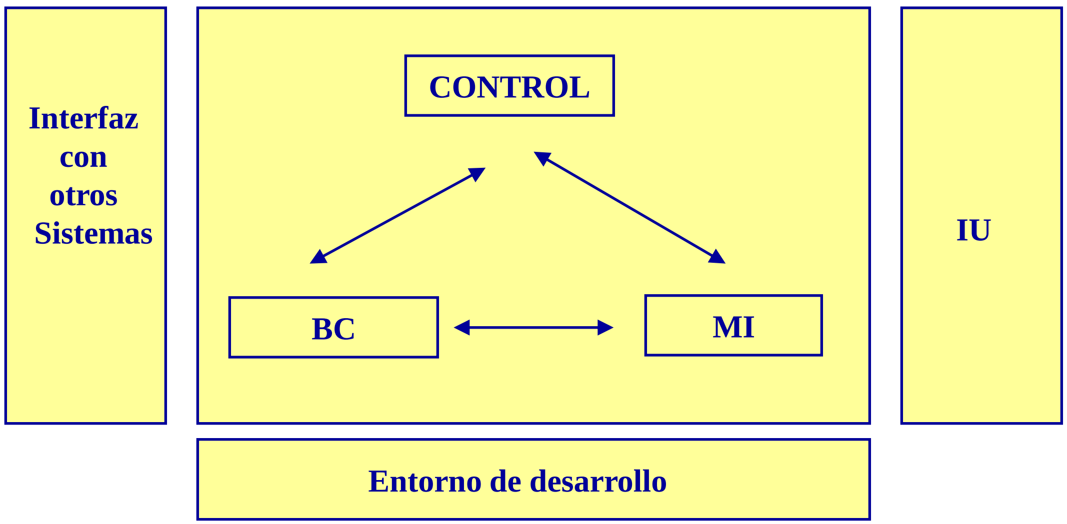
Representación de Conocimientos: Lógica formal y lógica descriptiva

Oscar Corcho García
(basado en transparencias de Asunción Gómez Pérez, Mariano Fernández López, Sean Bechhofer e Ian Horrocks)
ocorcho@fi.upm.es
Despacho 2107

Departamento de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain

Arquitectura de un Sistema Inteligente

- **Sistemas con los que interacciona**
- **Redes**
- **Bases de Datos**
- **...**



- **Hacer inferencias "visibles" a los usuarios**
- **Explicación**
- **Automático / manual**

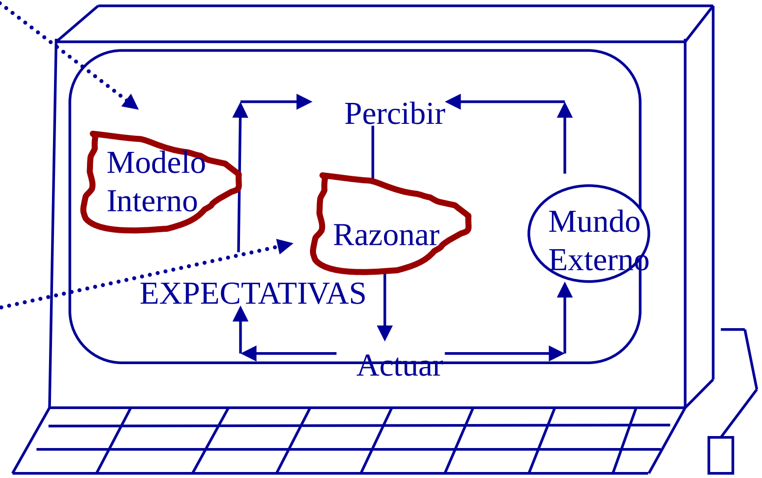
- **Herramientas de SBC**
- **Lenguajes de Programación**

Hipótesis Simbolista

El módulo de la BC del sistema está separado del módulo de razonamiento

Base de Conocimientos: Contienen conocimientos del dominio:

- **conceptos**
- **taxonomías**
- **relaciones "a medida" entre conceptos**
- **propiedades de conceptos**
- **hechos**
- **heurísticas**
- **Restricciones**
- **.....**

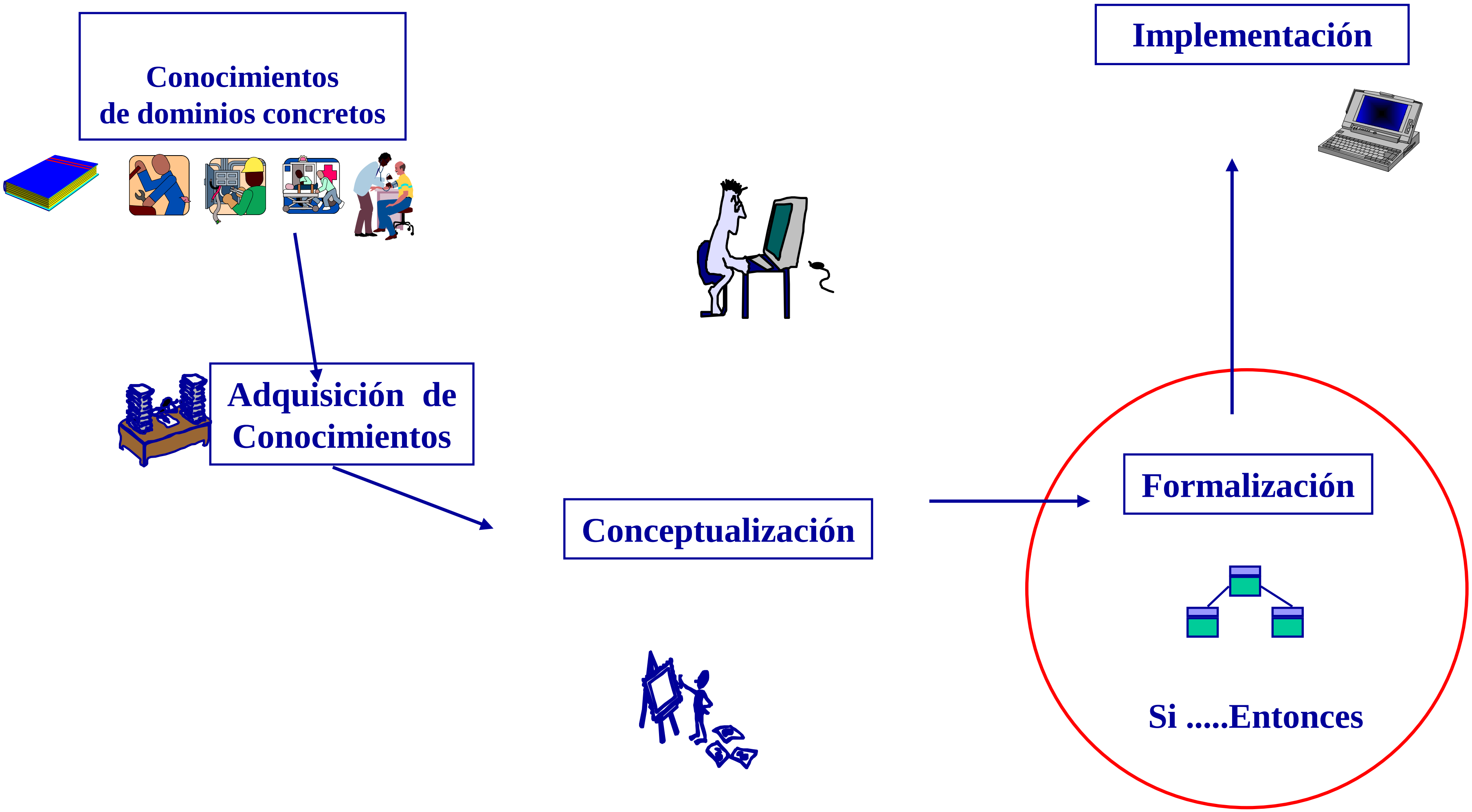


Motor de Inferencias:

•Permite que el sistema razone.

•A partir de los datos y conocimientos de entrada el sistema pueda producir una salida.

Escenario


Conocimientos
de dominios concretos
Implementación
Adquisición de
Conocimientos
Conceptualización
Formalización
Si .....Entonces

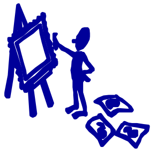
Pasos en el desarrollo de un SBC

Evaluación

Estudio Viabilidad

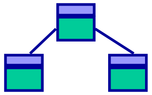
Adquisición Conocimientos

- **Entrevistas: abiertas, cerradas**
- **Análisis de Protocolos**
- **Emparrillado**
- **...**



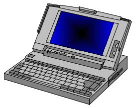
Conceptualización

- **Modelo Conceptual**
  - **Conocimientos Fácticos**
  - **Conocimientos Tácticos**
  - **Conocimientos Estratégicos**
  - **Metaconocimientos**



Formalización

- **Modelos Formal**
  - **Reglas**
  - **Redes Semánticas**
  - **Lógica**
  - **Marcos**
  - **Guiones**
  - **Restricciones**
- **Representar Conocimientos Inciertos**
  - **MYCIN**
  - **PROSPECTOR**
  - **Dempster Shaffer**
  - **Redes Bayesianas**
  - **Log. Fuzzy**
- **Técnicas de control**
  - **Metarreglas**
  - **Agendas**
  - **Pizarras**
  - **...**



Implementar

a) **Lenguaje de Programación**
- **C**
- **C++**
- **Modula**
- **....**

b) **Entorno o Herramienta**
- **KEE**
- **ART**
- **KAPPA**
- **....**

Formalización en lógica de predicados de primer orden

Oscar Corcho (basado en transparencias de Mariano Fernández López)
ocorcho@fi.upm.es
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn.
Boadilla del Monte, 28660. Madrid. Spain

Índice

Sintaxis (I)

1. Sintaxis
2. Significado
3. Formalización
4. Deducción
5. Implementación en PROLOG

Sintaxis versus Semántica

- **Sintaxis:**
  - **Símbolos que se utilizan para representar**
  - **Aspectos de Notación**
  - **Cada formalismo tiene su sintaxis**

- **Semántica:**
  - **Significado de lo que se ha representado utilizando una sintaxis determinada**

Sintaxis (II)

1. Términos

- Un símbolo de constante es un término
- Un símbolo de variable es un término
- Si f es un símbolo de función, y t1, t2, ..., tn son términos, entonces f(t1, t2, ..., tn) es un término

Ejemplo:

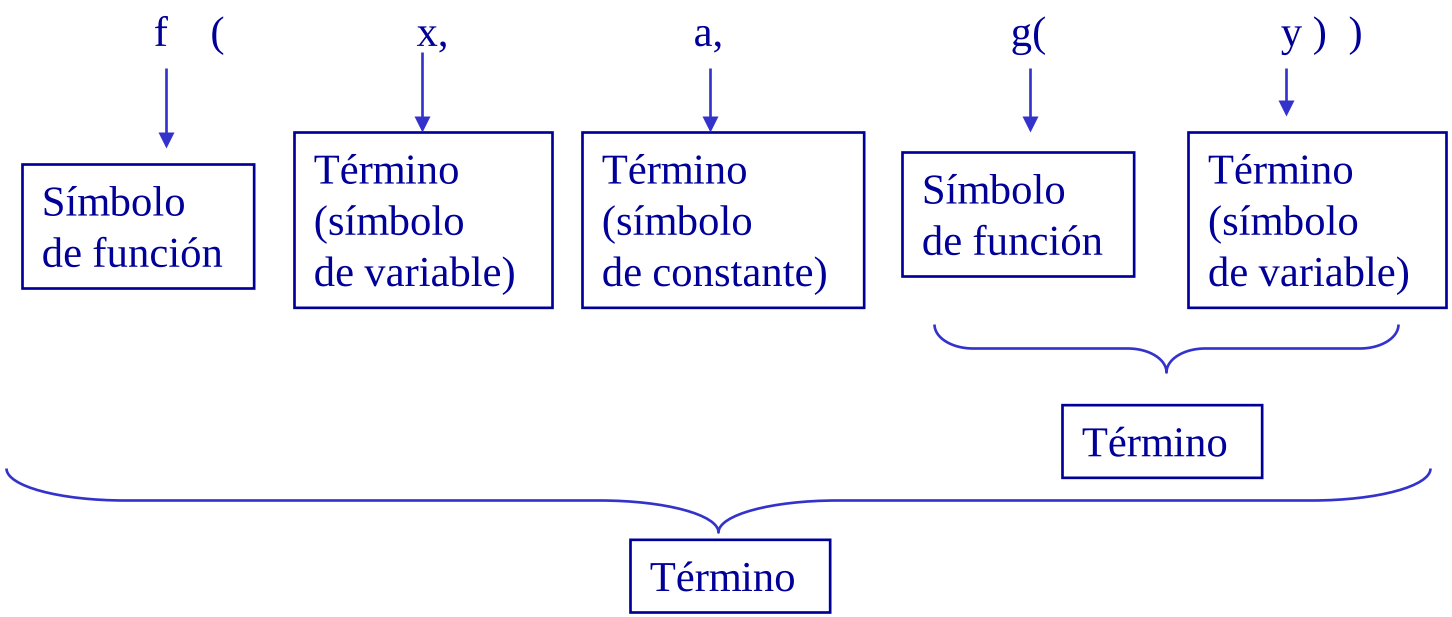
f (x, a, g(y))

Sintaxis (II)

1. Términos

- Un símbolo de constante es un término
- Un símbolo de variable es un término
- Si f es un símbolo de función, y t1, t2, ..., tn son términos, entonces f(t1, t2, ..., tn) es un término

Ejemplo:



Sintaxis (III)

2. Fórmulas

- Si p es un símbolo de predicado, y t1, t2, ..., tn son términos, entonces p(t1, t2, ..., tn) es una fórmula
- Si F es una fórmula, entonces $\neg$ F es una fórmula
- Si F y G son fórmulas, entonces:
  a) F $\wedge$ G es una fórmula
  b) F $\vee$ G es una fórmula
  c) F $\rightarrow$ G es una fórmula
  d) F $\leftrightarrow$ es una fórmula
- Si x es un símbolo de variable, y F es una fórmula, entonces:
  a) $\forall$ x F es una fórmula
  b) $\exists$ x F es una fórmula

Sintaxis (IV)

Ejemplo de fórmula

$\forall$ x (p(x) $\rightarrow$ q (x))

Sintaxis (IV)

Ejemplo de fórmula

$\forall$ x (p(x) $\rightarrow$ q (x))

Símbolo de variable

Símbolo de predicado

Término

Símbolo de predicado

Término

Fórmula

Fórmula

Fórmula

Fórmula

Significado (I)

1. Sintaxis
2. Significado
3. Formalización
4. Deducción
5. Implementación en PROLOG

Significado (II)

•Se puede establecer una correspondencia entre los símbolos lógicos y los objetos de un dominio (universo del discurso)

Constantes — *LPPO* | *CONJUNTOS* — Universo del discurso

•a
•b
•c

•Sócrates
•Platón
•Aristóteles

Símbolos de predicado

•h — hombre = {Sócrates, Platón, Aristóteles}

•m — mortal = {Sócrates, Platón, Aristóteles}

•t — maestroDe= {(Sócrates, Platón), (Platón, Aristóteles)}

Fórmulas

$\forall$ x (h (x) $\rightarrow$ m (x)) — mortal $\subset$ hombre

$\exists$ x $\exists$ y (h (x) $\wedge$ h (y) $\wedge$ t(x, y)) — maestroDe $\neq \emptyset$ en el dominio de los hombres

h (a) $\wedge$ h (b) $\wedge$ h(c) — Sócrates, Platón, Aristóteles $\in$ hombre

Significado (III)

•Los mismos símbolos pueden tener una correspondencia con objetos diferentes

LPPO	*CONJUNTOS*
Constantes	Universo del discurso
•a	•rotulador
•b	•borrador
•c	•lápiz
Símbolos de predicado	
•h	fungible = {rotulador, borrador, lápiz}
•m	material = {rotulador, borrador, lápiz}
•t	compradoAntes= {(rotulador, borrador), (borrador, lápiz)}
Fórmulas	
$\forall$ x (h (x) $\rightarrow$ m (x))	fungible $\subset$ material
$\exists$ x $\exists$ y (h (x) $\wedge$ h (y) $\wedge$ t(x, y))	compradoAntes $\neq \emptyset$ en el dominio del fungible
h (a) $\wedge$ h (b) $\wedge$ h(c)	rotulador, borrador, lápiz $\in$ fungible

Formalización (I)

1. Sintaxis

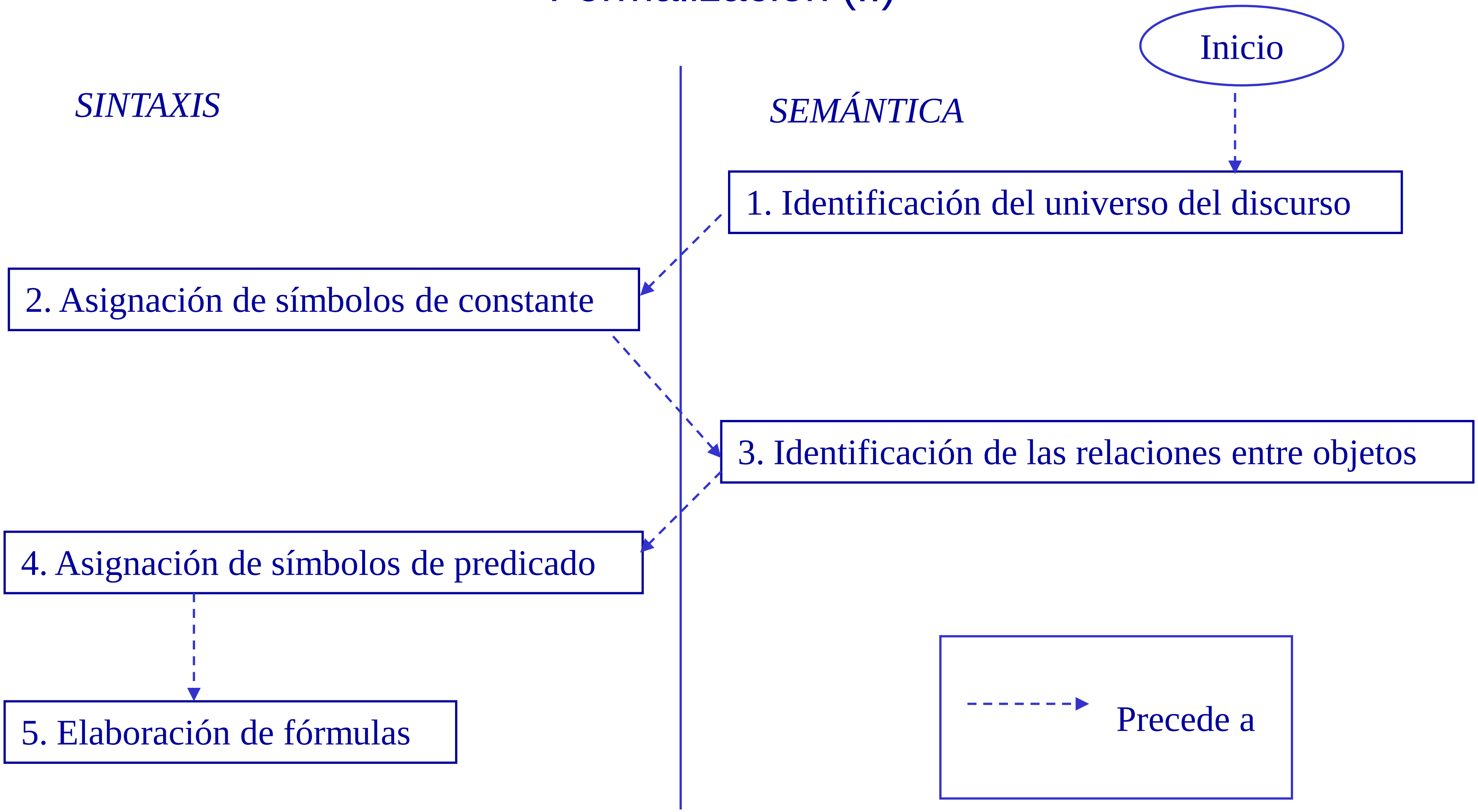
2. Significado

3. Formalización

4. Deducción

5. Implementación en PROLOG

Formalización (II)



Formalización (III)

Enunciado:

Se tiene un robot que distribuye paquetes en oficinas. Se sabe que:

- Los paquetes de la habitación 27 son más pequeños que los de la habitación 28.
- Todos los paquetes de la misma habitación son del mismo tamaño.
- En un instante concreto el robot sabe que:
    - i) El paquete A está en la habitación 27 ó 28 (pero no sabe en cuál).
    - ii) El paquete B está en la habitación 27.
    - iii) El paquete B no es más pequeño que el A.

El robot quiere probar que el paquete A está en la habitación 27.

Se pide:

a) Modelizar con lógica de predicados de primer orden.

Formalización (IV)

Enunciado:

Se tiene un robot que distribuye paquetes en oficinas. Se sabe que:

- Los paquetes de la habitación 27 son más pequeños que los de la habitación 28.
  - $\forall x \; \forall y \; (paquete(x) \wedge situadoEn\,(x,h27) \wedge paquete(y) \wedge situadoEn\,(y,h28) \rightarrow menor(x,y))$
- Todos los paquetes de la misma habitación son del mismo tamaño.
  - $\forall x \; \forall y \; \forall h \; (paquete(x) \wedge paquete(y) \wedge habitacion(h) \wedge situadoEn\,(x,h) \wedge situadoEn\,(y,h) \rightarrow igual(x,y))$
- En un instante concreto el robot sabe que:
  - i) El paquete A está en la habitación 27 ó 28 (pero no sabe en cuál).
    - $paquete(a) \wedge habitacion(h27) \wedge habitacion(h28) \wedge (situadoEn(a,h27) \vee situadoEn(a,h28))$
  - ii) El paquete B está en la habitación 27.
    - $paquete(b) \wedge situadoEn(b,h27)$
  - iii) El paquete B no es más pequeño que el A.
    - $\neg menor(b,a)$

El robot quiere probar que el paquete A está en la habitación 27.

¿$situadoEn(a,h27)$?

Deducción (I)

1. Sintaxis

2. Significado

3. Formalización

4. Deducción

5. Implementación en PROLOG

Deducción (II)

Se trata de saber si una fórmula Q es cierta conociendo:

1) Los axiomas lógicos, es decir, axiomas que son válidos sea cual sea el significado de los símbolos)

Por ejemplo, $\neg F \vee F$

2) Los axiomas no lógicos, es decir, los que son válidos sólo suponiendo ciertos significados de los símbolos.

3) Las reglas de inferencia

Por ejemplo (modus ponens)

$P \rightarrow Q$
P

Q

Deducción (III)

Una de las opciones a la hora de utilizar formas normales es las cláusulas de Horn

- ¿Qué es una cláusula? Es una disyunción de cualquier número de fórmulas atómicas afirmadas o negadas

- Las cláusulas de Horn se caracterizan por tener uno y sólo un átomo afirmado y cualquier número de átomos negados

  Por ejemplo:

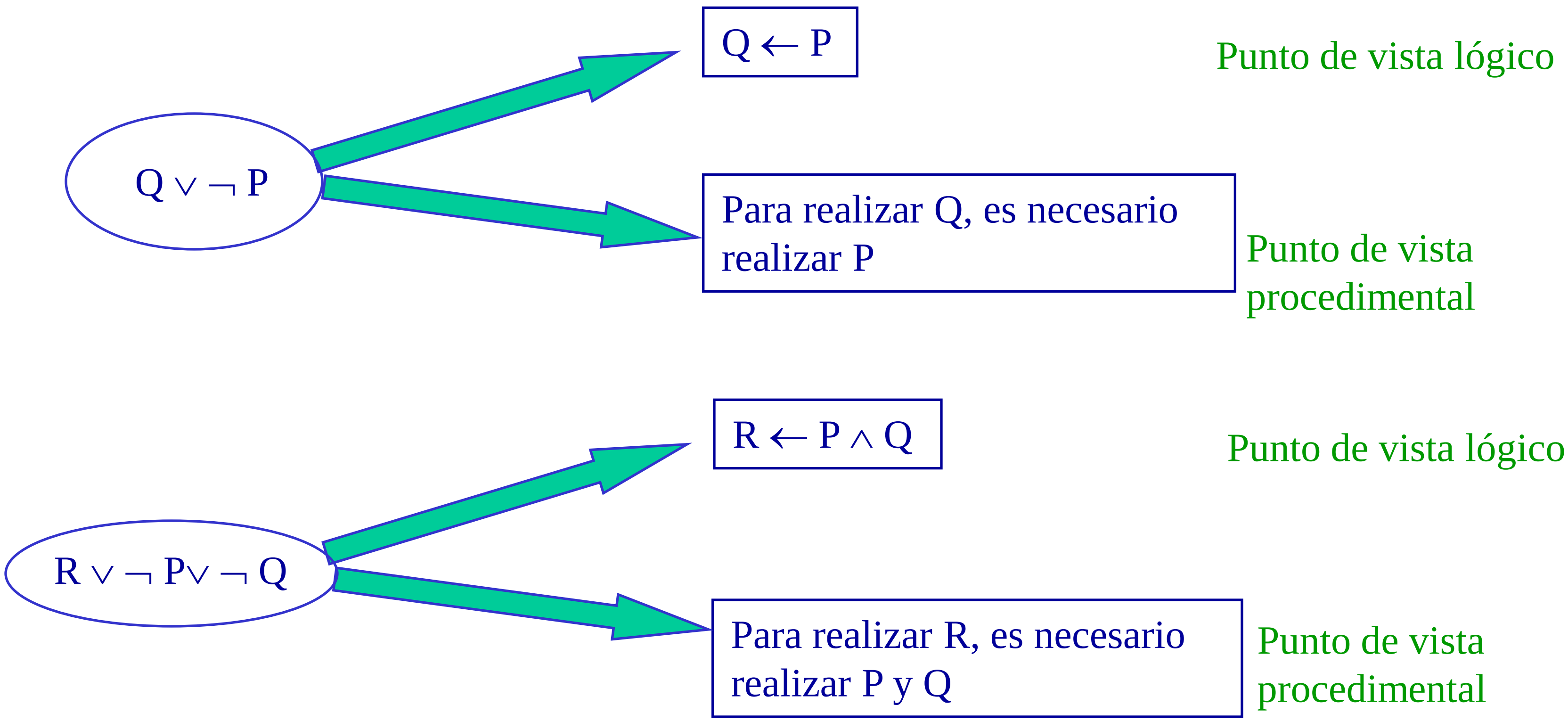
  P,

  $Q \lor \neg P$,

  $R \lor \neg P \lor \neg Q$

- No todas las fórmulas se pueden transformar en cláusulas de Horn

Deducción (IV)



Deducción (V)

Se puede hacer mediante resolución de cláusulas de Horn

Re-formulación del problema

F1, F2, ..., Fn, ¬ Q

Regla (resolución)

$$\begin{array}{c} P \vee Q \\ \neg P \vee R \\ \hline Q \vee R \end{array}$$

Se busca una contradicción:

□ que es la cláusula vacía
se obtiene de resolver F y ¬ F

Base de conocimientos

F1, F2, ..., Fn

Motor de inferencias

Interfaz de usuario

Consulta:
Q?

Sistema basado en conocimientos

Deducción (VI)

$\neg P \vee S$ $\neg S$ ---------- $\neg P$	Equivale a	$P \rightarrow S$ $\neg S$ ---------- $\neg P$ Modus tolens

Se hace un encadenamiento hacia atrás

Ejemplo:

•Base de conocimientos: $Q \vee \neg T$, $T \vee \neg S$, $S \vee \neg P$, P también se puede escribir como $Q \leftarrow T$, $T \leftarrow S$, $S \leftarrow P$, P

•Consulta: Q?

Deducción (VII)

$\neg P \vee S$ $\neg S$ ---------- $\neg P$	Equivale a	$P \rightarrow S$ $\neg S$ ---------- $\neg P$ Modus tolens

Se hace un encadenamiento hacia atrás

Ejemplo:

Base de conocimientos: $Q \leftarrow T$, $T \leftarrow S$, $S \leftarrow P$, P

Consulta: Q?

Paso 1. $Q \leftarrow T$
$\neg Q$

$\neg T$

Deducción (VII)

$$\begin{array}{l} \neg P \vee S \\ \neg S \\ \text{----------} \\ \neg P \end{array} \quad \text{Equivale a} \quad \begin{array}{l} P \rightarrow S \\ \neg S \\ \text{----------} \\ \neg P \end{array}$$

Modus tolens

Se hace un encadenamiento hacia atrás

Ejemplo:

Base de conocimientos: $Q \leftarrow T$, $T \leftarrow S$, $S \leftarrow P$, P

Consulta: Q?

Paso 1.

$$\begin{array}{l} Q \leftarrow T \\ \neg Q \\ \text{----------} \\ \neg T \end{array}$$

Paso 2.

$$\begin{array}{l} T \leftarrow S \\ \neg T \\ \text{----------} \\ \neg S \end{array}$$

Deducción (VII)

$\neg P \vee S$
$\neg S$

$\neg P$

Equivale a

$P \rightarrow S$
$\neg S$

$\neg P$

Modus tolens

Se hace un encadenamiento hacia atrás

Ejemplo:

Base de conocimientos: $Q \leftarrow T$, $T \leftarrow S$, $S \leftarrow P$, P
Consulta: Q?

Paso 1.	Paso 2.	Paso 3.
$Q \leftarrow T$ $\neg Q$ ---------- $\neg T$	$T \leftarrow S$ $\neg T$ ---------- $\neg S$	$S \leftarrow P$ $\neg S$ ---------- $\neg P$

Deducción (VII)

$\neg P \vee S$ $\neg S$ ---------- $\neg P$	Equivale a	$P \rightarrow S$ $\neg S$ ---------- $\neg P$ Modus tolens

Se hace un encadenamiento hacia atrás

Ejemplo:

Base de conocimientos: $Q \leftarrow T$, $T \leftarrow S$, $S \leftarrow P$, P

Consulta: Q?

Paso 1. $Q \leftarrow T$ $\neg Q$ ---------- $\neg T$	Paso 2. $T \leftarrow S$ $\neg T$ ---------- $\neg S$	Paso 3. $S \leftarrow P$ $\neg S$ ---------- $\neg P$	Paso 4. P $\neg P$ ---------- $\square$

Deducción (VIII)

Si la formalización se ha realizado en lógica de predicados de primer orden, entonces, en la implementación, además de resolución, también hay que aplicar sustitución (e.g.)

$\neg$ Q(x) $\vee$ T(x)
$\neg$ Q(a)

$\neg$ T(a)

Ejercicio de formalización y deducción

a) Formalizar el siguiente texto en lógica de primer orden

> "El que no estudia una asignatura no aprueba su examen"
> "Hay alumnos que además de no estudiar ninguna asignatura tienen mala suerte en el examen de Inteligencia Artificial"
> "El que estudia una asignatura y no se pone nervioso en su examen, lo aprueba a no ser que tenga mala suerte en su examen"
> "Juan ha aprobado Inteligencia Artificial"
> "Luego Juan ha estudiado Inteligencia Artificial"

b) Comprobar si la estructura deductiva anterior es correcta utilizando el método de resolución

Implementación en PROLOG (I)

1. Sintaxis

2. Significado

3. Formalización

4. Deducción

5. Implementación en PROLOG

Implementación en PROLOG (I)

Se basa en la formalización en cláusulas de Horn:

Ejemplo:

Las cláusulas de la forma

P,
$Q \vee \neg P$,
$R \vee \neg P \vee \neg Q$

se escriben en PROLOG como

```
p.
q :- p,
r:-  p, q
```

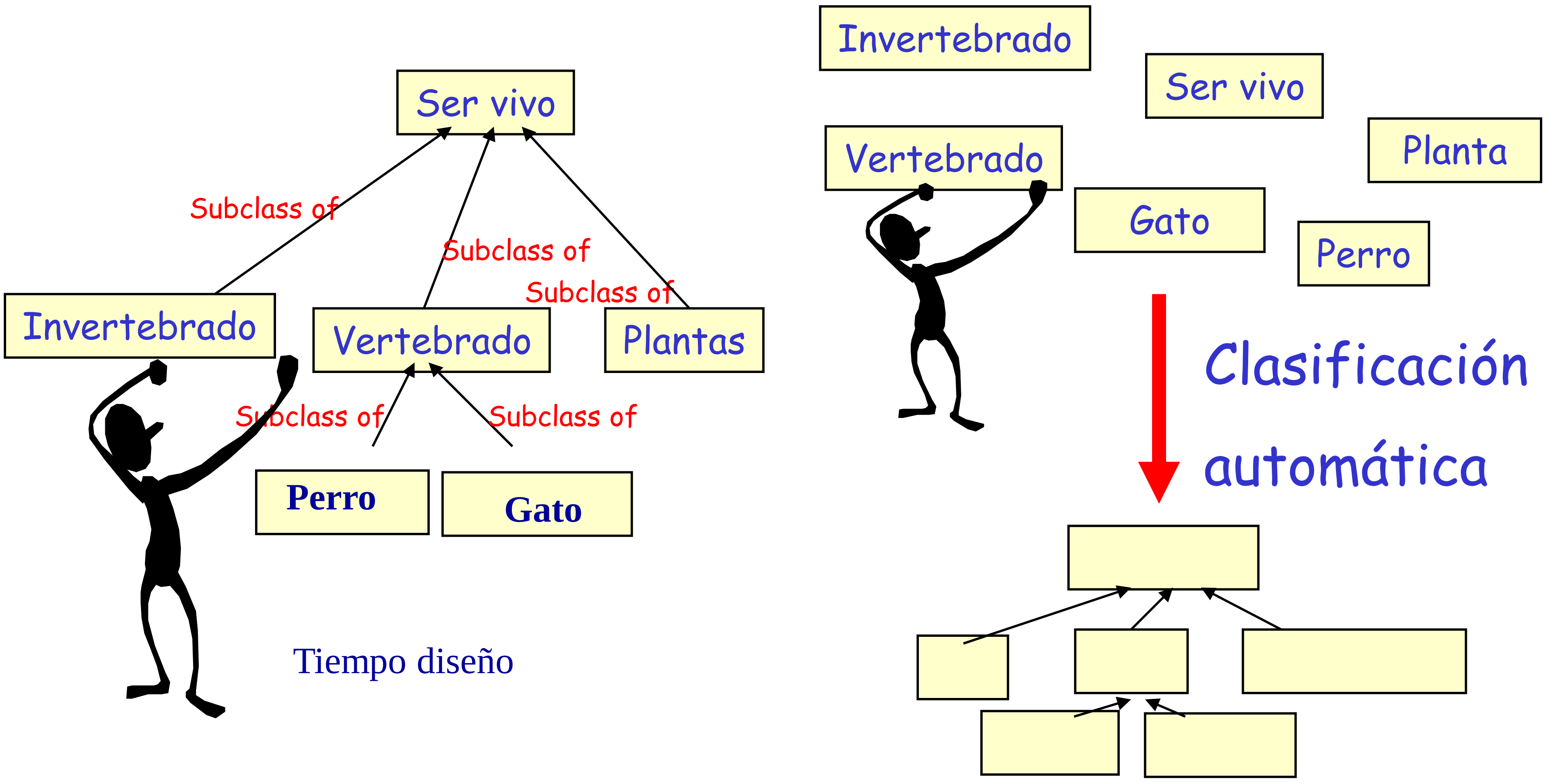
Hay recursos para atenuar el inconveniente de que no todas
las fórmulas lógicas se pueden expresar como cláusulas de Horn

Formalización en lógica descriptiva

Oscar Corcho (basado en transparencias de Sean Bechhofer e Ian Horrocks)
ocorcho@fi.upm.es
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn.
Boadilla del Monte, 28660. Madrid. Spain

Lógicas Descriptivas

- **Clasificación automática realizada por el motor de inferencias del lenguaje**
- **En tiempo de Ejecución**

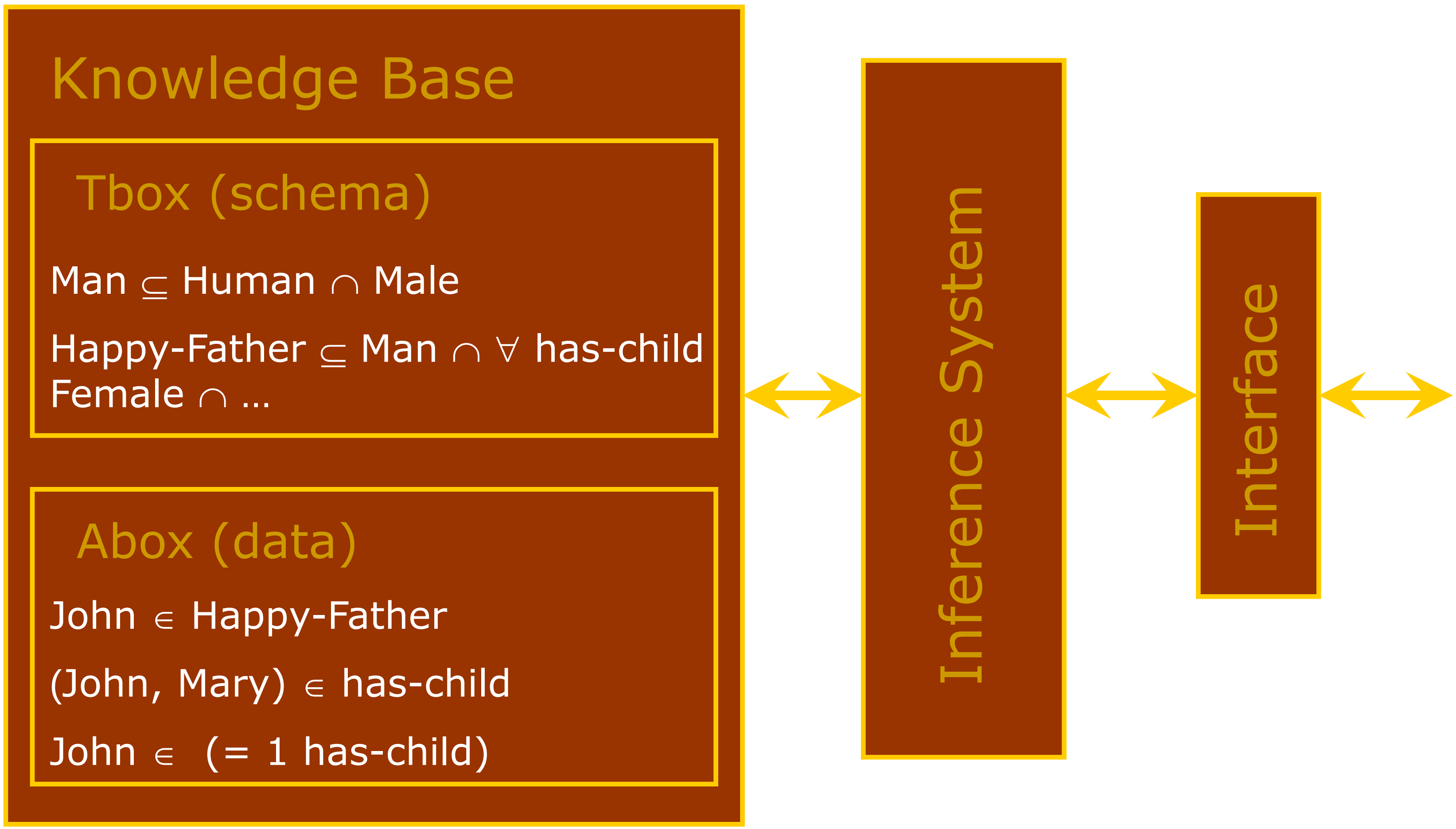


What is Description Logic?

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals
    - Specific languages characterised by the constructors and axioms used to assert knowledge about classes, roles and individuals.
    - Example: ALC (the least expressive language in DL that is propositionally closed)
      - Constructors: boolean (and, or, not)
      - Role restrictions

- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimised)

DL Architecture



<table>
<tr><th>Construct</th><th>Syntax</th><th colspan="4">Language</th></tr>
<tr><td>Concept</td><td>A</td><td rowspan="4">FL_0</td><td rowspan="7">FL^-</td><td rowspan="8">AL</td><td rowspan="11">S[14]</td></tr>
<tr><td>Role name</td><td>R</td></tr>
<tr><td>Intersection</td><td>$C \cap D$</td></tr>
<tr><td>Value restriction</td><td>$\forall R.C$</td></tr>
<tr><td>Limited existential quantification</td><td>$\exists R$</td><td></td></tr>
<tr><td>Top or Universal</td><td>$\top$</td><td></td></tr>
<tr><td>Bottom</td><td>$\perp$</td><td></td></tr>
<tr><td>Atomic negation</td><td>$\neg A$</td><td></td><td></td></tr>
<tr><td>Negation[15]</td><td>$\neg C$</td><td colspan="3">C</td></tr>
<tr><td>Union</td><td>$C \cup D$</td><td colspan="3">U</td></tr>
<tr><td>Existential restriction</td><td>$\exists R.C$</td><td colspan="3">E</td></tr>
<tr><td>Number restrictions</td><td>$(\geq n\ R)\ \ (\leq n\ R)$</td><td colspan="3">N</td><td></td></tr>
<tr><td>Nominals</td><td>$\{a_1 \ldots a_n\}$</td><td colspan="3">O</td><td></td></tr>
<tr><td>Role hierarchy</td><td>$R \subseteq S$</td><td colspan="3">H</td><td></td></tr>
<tr><td>Inverse role</td><td>R^-</td><td colspan="3">I</td><td></td></tr>
<tr><td>Qualified number restriction</td><td>$(\geq n\ R.C)\ \ (\leq n\ R.C)$</td><td colspan="3">Q</td><td></td></tr>
</table>

→ ≥3 hasChild, ≤1 hasMother

→ {Colombia, Argentina, México, ...} → MercoSur countries

→ hasChild⁻ (hasParent)

→ ≤2 hasChild.Female, ≥1 hasParent.Male

Other:

Concrete datatypes: hasAge.(<21)

Transitive roles: hasChild* (descendant)

Role composition: hasParent o hasBrother (uncle)

---

[12] Names previously used for Description Logics were: terminological knowledge representation languages, concept languages, term subsumption languages, and KL-ONE-based knowledge representation languages.

[13] In this table, we use A to refer to atomic concepts (concepts that are the basis for building other concepts), C and D to any concept definition, R to atomic roles and S to role definitions. FL is used for structural DL languages and AL for attributive languages (Baader et al., 2003).

[14] S is the name used for the language ALC_{R+}, which is composed of ALC plus transitive roles.

[15] ALC and ALCUE are equivalent languages, since union (U) and existential restriction (E) can be represented using negation (C).

Semantics. Class constructors

Constructor	DL Syntax	Example	Modal Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	$\neg$Male	$\neg C$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}	$x_1 \vee \ldots \vee x_n$
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leqslant n P$	$\leqslant$1hasChild	$[P]_{n+1}$
minCardinality	$\geqslant n P$	$\geqslant$2hasChild	$\langle P \rangle_n$

- XML Schema datatypes are treated as classes
  - $\forall$hasAge.nonNegativeInteger
- Nesting of constructors can be arbitrarily complex
  - Person $\wedge$ $\forall$hasChild.(Doctor $\vee$ $\exists$hasChild.Doctor)
- Lots of redundancy, e.g., use negations to transform and to or and exists to forall

Semantics. OWL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ $\neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

- Axioms (mostly) reducible to inclusion (v)
  - C ≡ D iff both C ⊆ D and D ⊆ C
  - C disjoint D iff C ⊆ ¬D

Formalisation. Some basic DL modelling guidelines

• X must be Y, X is an Y that...	→ $X \subseteq Y$
• X is exactly Y, X is the Y that...	→ $X \equiv Y$
• X is not Y (*not the same as X is whatever it is not Y*)	→ $X \subseteq \neg Y$
• X and Y are disjoint	→ $X \cap Y \subseteq \perp$
• X is Y or Z	→ $X \subseteq Y \cup Z$
• X is Y for which property P has only instances of Z as values	→ $X \subseteq Y \cap (\forall P.Z)$
• X is Y for which property P has at least an instance of Z as a value	→ $X \subseteq Y \cap (\exists P.Z)$
• X is Y for which property P has at most 2 values	→ $X \subseteq Y \cap (\leq 2.P)$
• Individual X is a Y	→ $X \in Y$

Chunk 1. Formalize in DL, and then in OWL DL

1. Concept definitions:

   Grass and trees must be plants. Leaves are parts of a tree but there are other parts of a tree that are not leaves. A dog must eat bones, at least. A sheep is an animal that must only eat grass. A giraffe is an animal that must only eat leaves. A mad cow is a cow that eats brains that can be part of a sheep.

2. Restrictions:

   Animals or part of animals are disjoint with plants or parts of plants.

3. Properties:

   Eats is applied to animals. Its inverse is eaten_by.

4. Individuals:

   Tom.
   Flossie is a cow.
   Rex is a dog and is a pet of Mick.
   Fido is a dog.
   Tibbs is a cat.

Chunk 2. Formalize in DL, and then in OWL DL

1. Concept definitions:

   Bicycles, buses, cars, lorries, trucks and vans are vehicles. There are several types of companies: bus companies and haulage companies.

   An elderly person must be adult. A kid is (exactly) a person who is young. A man is a person who is male and is adult. A woman is a person who is female and is adult. A grown up is a person who is an adult. And old lady is a person who is elderly and female. Old ladies must have some animal as pets and all their pets are cats.

2. Restrictions:

   Youngs are not adults, and adults are not youngs.

3. Properties:

   Has mother and has father are subproperties of has parent.

4. Individuals:

   Kevin is a person.
   Fred is a person who has a pet called Tibbs.
   Joe is a person who has at most one pet. He has a pet called Fido.
   Minnie is a female, elderly, who has a pet called Tom.

Chunk 3. Formalize in DL, and then in OWL DL

1. Concept definitions:

    A magazine is a publication. Broadsheets and tabloids are newspapers. A quality broadsheet is a type of broadsheet. A red top is a type of tabloid. A newspaper is a publication that must be either a broadsheet or a tabloid.

    White van mans must read only tabloids.

2. Restrictions:

    Tabloids are not broadsheets, and broadsheets are not tabloids.

3. Properties:

    The only things that can be read are publications.

4. Individuals:

    Daily Mirror

    The Guardian and The Times are broadsheets

    The Sun is a tabloid

Chunk 4. Formalize in DL, and then in OWL DL

1. Concept definitions:
   A pet is a pet of something. An animal must eat something. A vegetarian is an animal that does not eat animals nor parts of animals. Ducks, cats and tigers are animals. An animal lover is a person who has at least three pets. A pet owner is a person who has animal pets. A cat liker is a person who likes cats. A cat owner is a person who has cat pets. A dog liker is a person who likes dogs. A dog owner is a person who has dog pets.

2. Restrictions:
   Dogs are not cats, and cats are not dogs.

3. Properties:
   Has pet is defined between persons and animals. Its inverse is is_pet_of.

4. Individuals:
   Dewey, Huey, and Louie are ducks.
   Fluffy is a tiger.
   Walt is a person who has pets called Huey, Louie and Dewey.

Chunk 5. Formalize in DL, and then in OWL DL

1. Concept definitions

   A driver must be adult. A driver is a person who drives vehicles. A lorry driver is a person who drives lorries. A haulage worker is who works for a haulage company or for part of a haulage company. A haulage truck driver is a person who drives trucks ans works for part of a haulage company. A van driver is a person who drives vans. A bus driver is a person who drives buses. A white van man is a man who drives white things and vans.

2. Restrictions:

   --

3. Properties:

   The service number is an integer property with no restricted domain

4. Individuals:

   Q123ABC is a van and a white thing.
   The42 is a bus whose service number is 42.
   Mick is a male who read Daily Mirror and drives Q123ABC.

Chunk 1. Formalisation in DL

$$grass \subseteq plant$$

$$tree \subseteq plant$$

$$leaf \subseteq \exists partOf.tree$$

$$dog \subseteq \exists eats.bone$$

$$sheep \subseteq animal \cap \forall eats.grass$$

$$giraffe \subseteq animal \cap \forall eats.leaf$$

$$madCow \equiv cow \cap \exists eats.(brain \cap \exists partOf.sheep)$$

$$(animal \cup \exists partOf.animal) \cap (plant \cup \exists partOf.plant) \subseteq \perp$$

Chunk 2. Formalisation in DL

$bicycle \subseteq vehicle; bus \subseteq vehicle; car \subseteq vehicle; lorry \subseteq vehicle; truck \subseteq vehicle$

$busCompany \subseteq company; haulageCompany \subseteq company$

$elderly \subseteq person \cap adult$

$kid \equiv person \cap young$

$man \equiv person \cap male \cap adult$

$woman \equiv person \cap female \cap adult$

$grownUp \equiv person \cap adult$

$oldLady \equiv person \cap female \cap elderly$

$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$

$young \cap adult \subseteq \perp$

$hasMother \subseteq hasParent$

$hasFather \subseteq hasParent$

Chunk 3. Formalisation in DL

$$magazine \subseteq publication$$

$$broadsheet \subseteq newspaper$$

$$tabloid \subseteq newspaper$$

$$qualityBroadsheet \subseteq broadsheet$$

$$redTop \subseteq tabloid$$

$$newspaper \subseteq publication \cap (broadsheet \cup tabloid)$$

$$whiteVanMan \subseteq \forall reads.tabloid$$

$$tabloid \cap broadsheet \subseteq \perp$$

Chunk 4. Formalisation in DL

$$pet \equiv \exists isPetOf.\mathrm{T}$$

$$animal \subseteq \exists eats.\mathrm{T}$$

$$vegetarian \equiv animal \cap \forall eats.\neg animal \cap \forall eats.\neg(\exists partOf.animal)$$

$$duck \subseteq animal; cat \subseteq animal; tiger \subseteq animal$$

$$animalLover \equiv person \cap (\geq 3 hasPet)$$

$$petOwner \equiv person \cap \exists hasPet.animal$$

$$catLike \equiv person \cap \exists likes.cat; catOwner \equiv person \cap \exists hasPet.cat$$

$$dogLike \equiv person \cap \exists likes.dog; dogOwner \equiv person \cap \exists hasPet.dog$$

$$dog \cap cat \subseteq \perp$$

Chunk 5. Formalisation in DL

$$driver \subseteq adult$$

$$driver \equiv person \cap \exists drives.vehicle$$

$$lorryDriver \equiv person \cap \exists drives.lorry$$

$$haulageWorke \equiv \exists worksFor.(haulageCompany \cup \exists partOf.haulageCompany)$$

$$haulageTruckDriver \equiv person \cap \exists drives.truck \cap$$
$$\exists worksFor.(\exists partOf.haulageCompany)$$

$$vanDriver \equiv person \cap \exists drives.van$$

$$busDriver \equiv person \cap \exists drives.bus$$

$$whiteVanMan \equiv man \cap \exists drives.(whiteThing \cap van)$$

Inference. Basic Inference Tasks

- Subsumption – check knowledge is correct (captures intuitions)
  - Does C subsume D w.r.t. ontology O? (in *every* model I of O, $C^I \subseteq D^I$)
- Equivalence – check knowledge is minimally redundant (no unintended synonyms)
  - Is C equivalent to D w.r.t. O? (in *every* model I of O, $C^I = D^I$)
- Consistency – check knowledge is meaningful (classes can have instances)
  - Is C satisfiable w.r.t. O? (there exists *some* model I of O s.t. $C^I \neq \emptyset$)
- Instantiation and querying
  - Is x an instance of C w.r.t. O? (in *every* model I of O, $x^I \in C^I$)
  - Is (x,y) an instance of R w.r.t. O? (in *every* model I of O, $(x^I,y^I) \in R^I$)
- All reducible to KB satisfiability or concept satisfiability w.r.t. a KB
- Can be decided using highly optimised tableaux reasoners

Interesting results (I). Automatic classification

And old lady is a person who is elderly and female.
Old ladies must have some animal as pets and all their pets are cats.

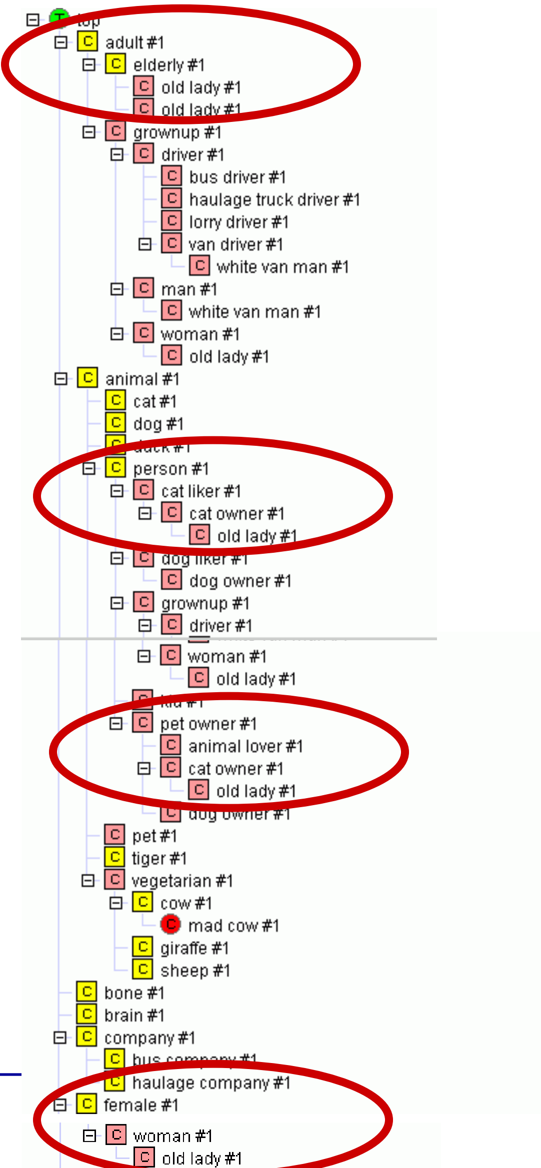
$$elderly \subseteq person \cap adult$$

$$woman \equiv person \cap female \cap adult$$

$$catOwner \equiv person \cap \exists hasPet.cat$$

$$oldLady \equiv person \cap female \cap elderly$$

$$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$$



We obtain:

Old ladies must be women.
Every old lady must have a pet cat
Hence, every old lady must be a cat owner

$$oldLady \subseteq woman \cap elderly \cap catOwner$$

Interesting results (II). Instance classification

A pet owner is a person who has animal pets
Old ladies must have some animal as pets and all their pets are cats.
Has pet has domain person and range animal
Minnie is a female, elderly, who has a pet called Tom.

$$petOwner \equiv person \cap \exists hasPet.animal$$

$$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$$

$$hasPet \subseteq (person, animal)$$

$$Minnie \in female \cap elderly$$

$$hasPet(Minnie, Tom)$$

We obtain:

Minnie is a person
Hence, Minnie is an old lady
Hence, Tom is a cat

$$Minnie \in person; Tom \in animal$$

$$Minnie \in petOwner$$

$$Minnie \in oldLady$$

$$Tom \in cat$$

Interesting results (III). Instance classification and redundancy detection

An animal lover is a person who has at least three pets
Walt is a person who has pets called Huey, Louie and Dewey.

$$animalLover \equiv person \cap (\geq 3 hasPet)$$

$$Walt \in person$$

$$hasPet(Walt, Huey)$$

$$hasPet(Walt, Louie)$$

$$hasPet(Walt, Dewey)$$

We obtain:

Walt is an animal lover
Walt is a person is redundant

$$Walt \in animalLover$$

Interesting results (IV). Instance classification

A van is a type of vehicle
A driver must be adult
A driver is a person who drives vehicles
A white van man is a man who drives vans and white things
White van mans must read only tabloids
Q123ABC is a white thing and a van
Mick is a male who reads Daily Mirror and drives Q123ABC

$$van \subseteq vehicle$$

$$driver \subseteq adult$$

$$driver \equiv person \cap \exists drives.vehicle$$

$$whiteVanMan \equiv man \cap \exists drives.(van \cap whiteThing)$$

$$whiteVanMan \subseteq \forall reads.tabloid$$

$$Q123ABC \in whiteThing \cap van$$

$$Mick \in male$$

$$reads(Mick, DailyMirror)$$

$$drives(Mick, Q123ABC)$$

We obtain:

Mick is an adult
Mick is a white van man
Daily Mirror is a tabloid

$$Mick \in adult$$

$$Mick \in whiteVanMan$$

$$DailyMirror \in tabloid$$

Interesting results (V). Consistency checking

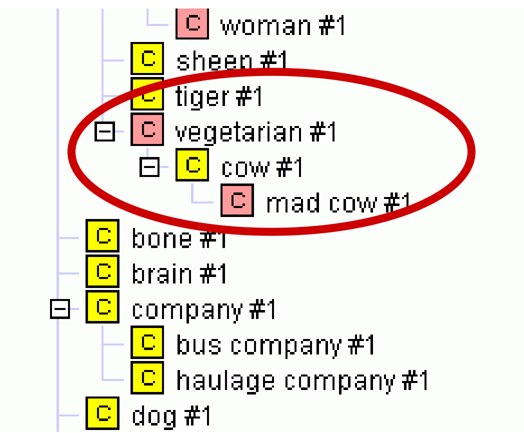
Cows are vegetarian.
A vegetarian is an animal that does not eat animals nor parts of animals.
A mad cow is a cow that eats brains that can be part of a sheep

$$cow \subseteq vegetarian$$

$$vegetarian \equiv animal \cap \forall eats.\neg animal \cap \forall eats.\neg(\exists partOf.animal))$$

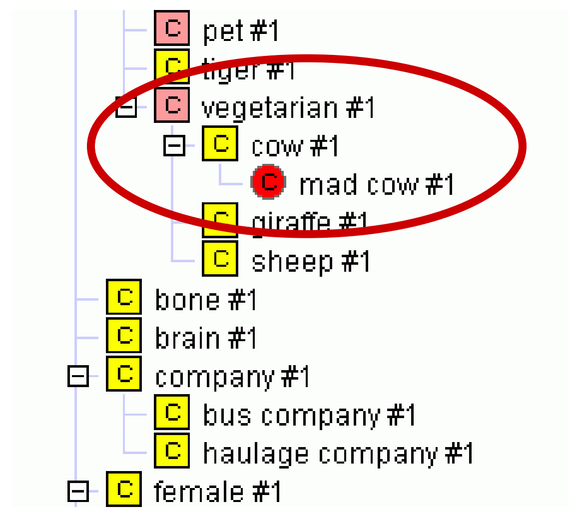
$$madCow \equiv cow \cap \exists eats.(brain \cup \exists partOf.sheep)$$

$$(animal \cup \exists partOf.animal) \cap (plant \cup \exists partOf.plant) \subseteq \perp$$



We obtain:

Mad cow is unsatisfiable



Tableaux Algorithms

- Try to prove satisfiability of a knowledge base
- How do they work
  - They try to build a model of input concept C
    - Tree model property
      - If there is a model, then there is a tree shaped model
    - If no tree model can be found, then input concept unsatisfiable
  - Decompose C syntactically
    - Work on concepts in negation normal form (De Morgan's laws)
    - Use of tableaux expansion rules
    - If non-deterministic rules are applied, then there is search
  - Stop (and backtrack) if clash
    - E.g. A(x), ¬A(x)
  - Blocking (cycle check) ensures termination for more expressive logics

- The algorithm finishes when no more rules can be applied or a conflict is detected

$x \bullet \{C_1 \sqcap C_2, \dots\}$	$\rightarrow_{\sqcap}$	$x \bullet \{C_1 \sqcap C_2, C_1, C_2, \dots\}$
$x \bullet \{C_1 \sqcup C_2, \dots\}$	$\rightarrow_{\sqcup}$	$x \bullet \{C_1 \sqcup C_2, C, \dots\}$ for $C \in \{C_1, C_2\}$
$x \bullet \{\exists R.C, \dots\}$	$\rightarrow_{\exists}$	$x \bullet \{\exists R.C, \dots\}$ $R \downarrow$ $y \bullet \{C\}$
$x \bullet \{\forall R.C, \dots\}$ $R \downarrow$ $y \bullet \{\dots\}$	$\rightarrow_{\forall}$	$x \bullet \{\forall R.C, \dots\}$ $R \downarrow$ $y \bullet \{C, \dots\}$
$x \bullet \{\forall R.C, \dots\}$ $R \downarrow$ $y \bullet \{\dots\}$	$\rightarrow_{\forall_+}$	$x \bullet \{\forall R.C, \dots\}$ $R \downarrow$ $y \bullet \{\forall R.C, \dots\}$

Tableaux examples and exercises

- Example
  - $\exists S.C \wedge \forall S.(\neg C \vee \neg D) \wedge \exists R.C \wedge \forall R.(\exists R.C)$
- Exercise 1
  - $\exists R.(\exists R.D) \wedge \exists S.\neg D \wedge \forall S.(\exists R.D)$
- Exercise 2
  - $\exists R.(C \vee D) \wedge \forall R.\neg C \wedge \neg\exists R.D$

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$$

$$w$$

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$$

w

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$

w

Tableaux Algorithm — Example

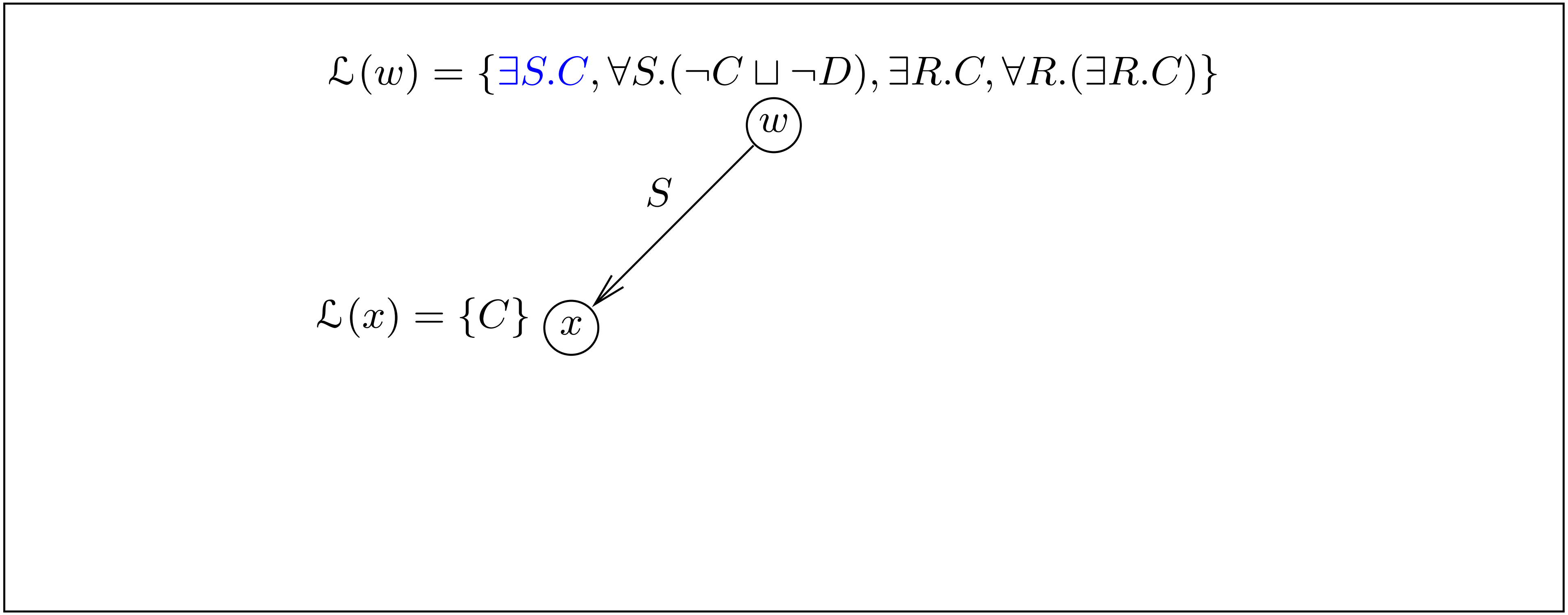
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$

$$w$$

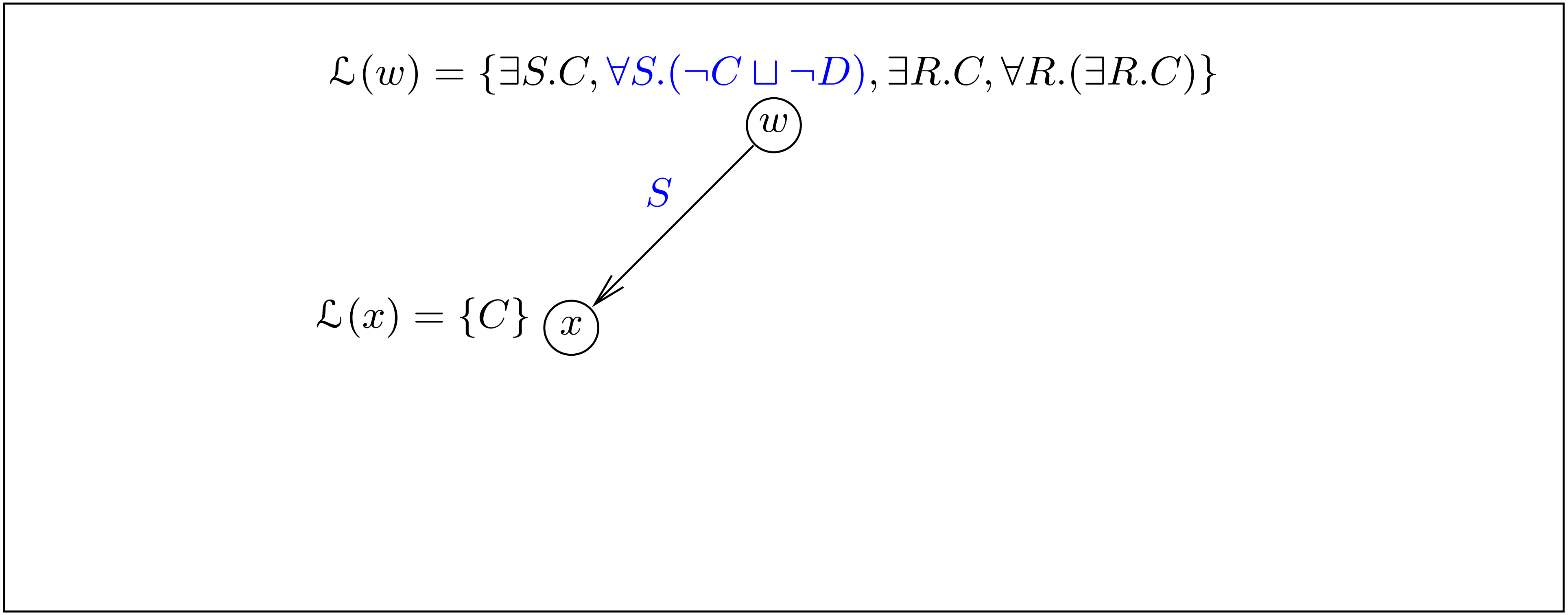
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



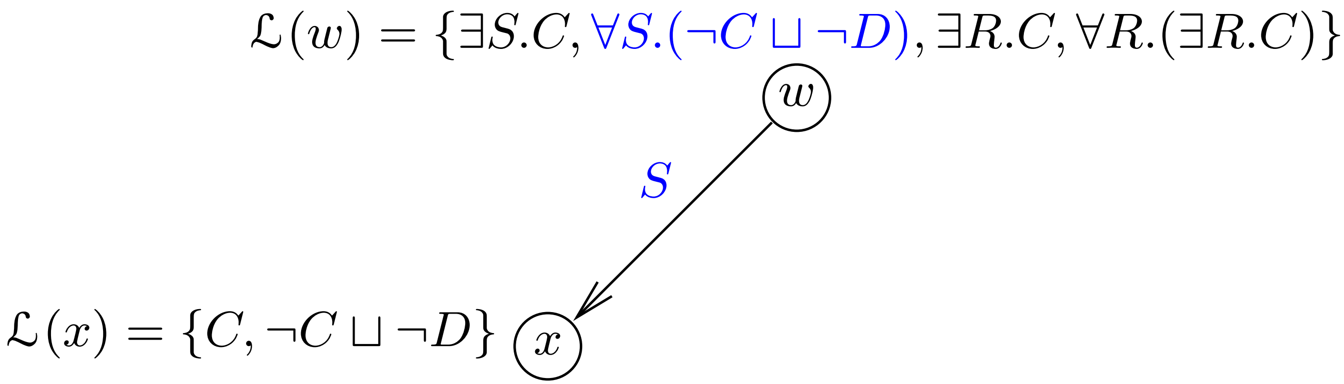
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



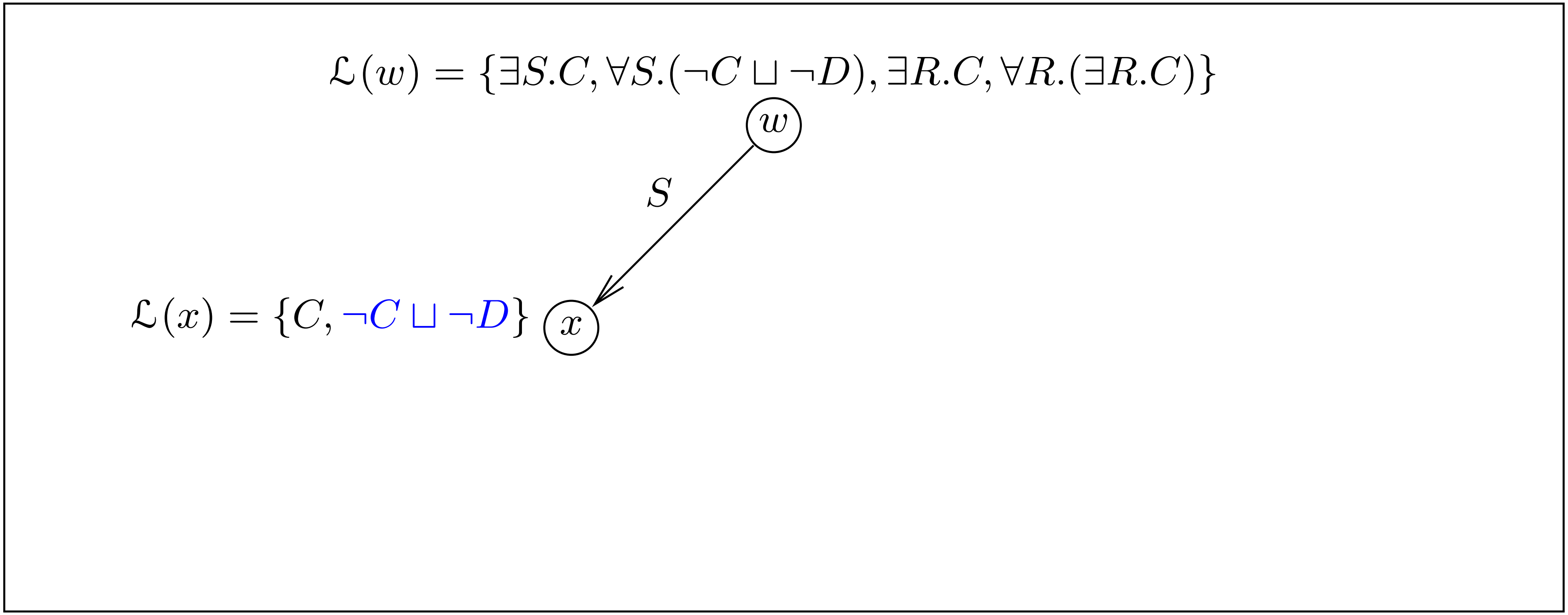
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



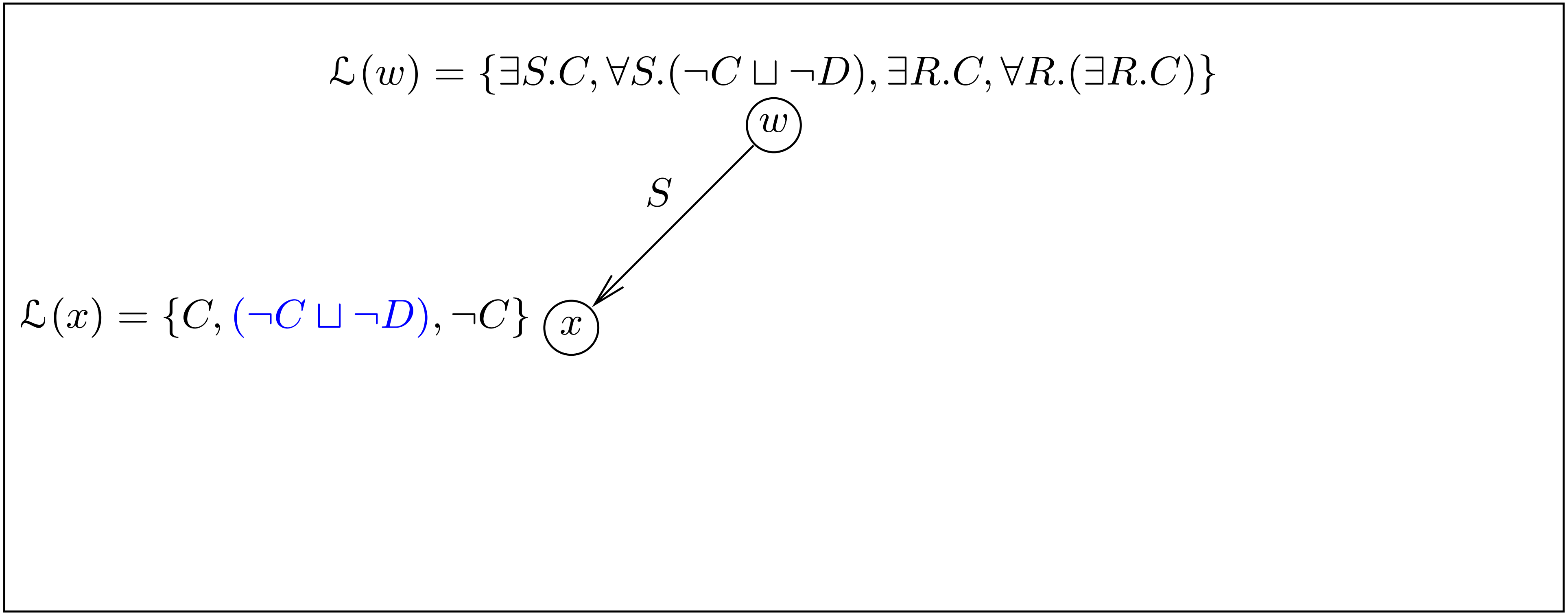
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



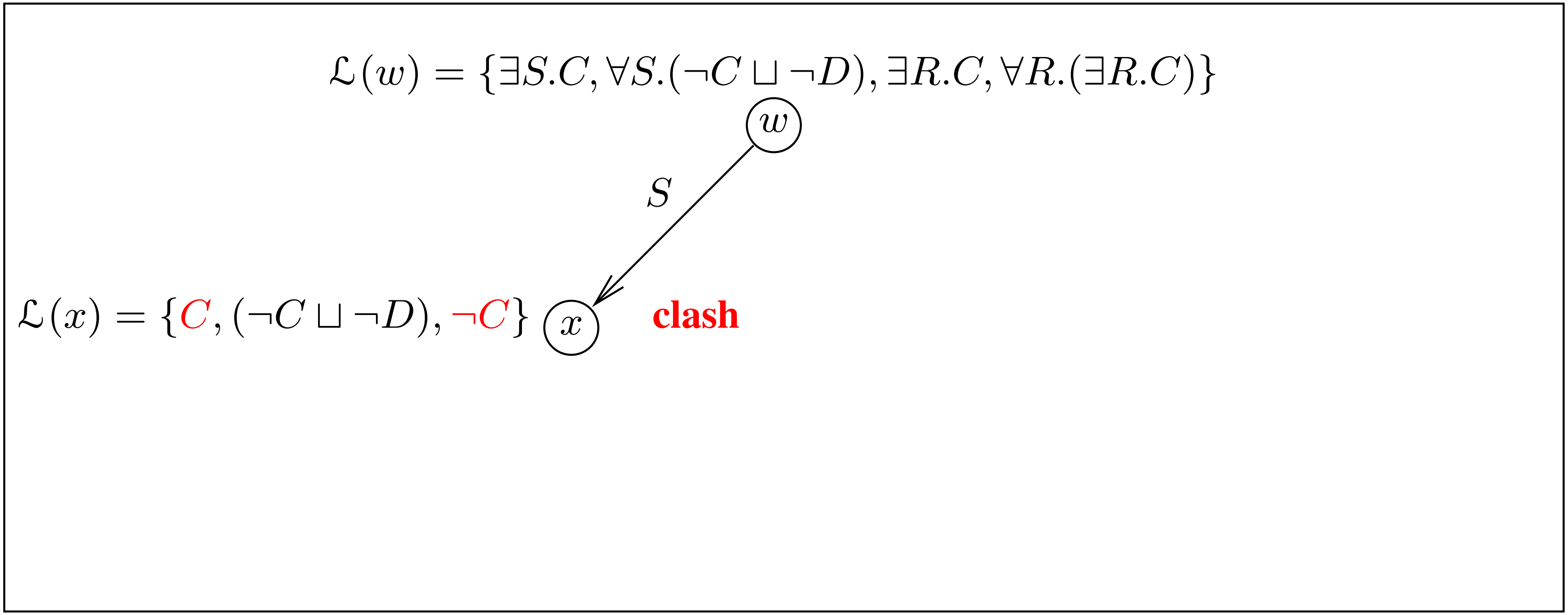
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



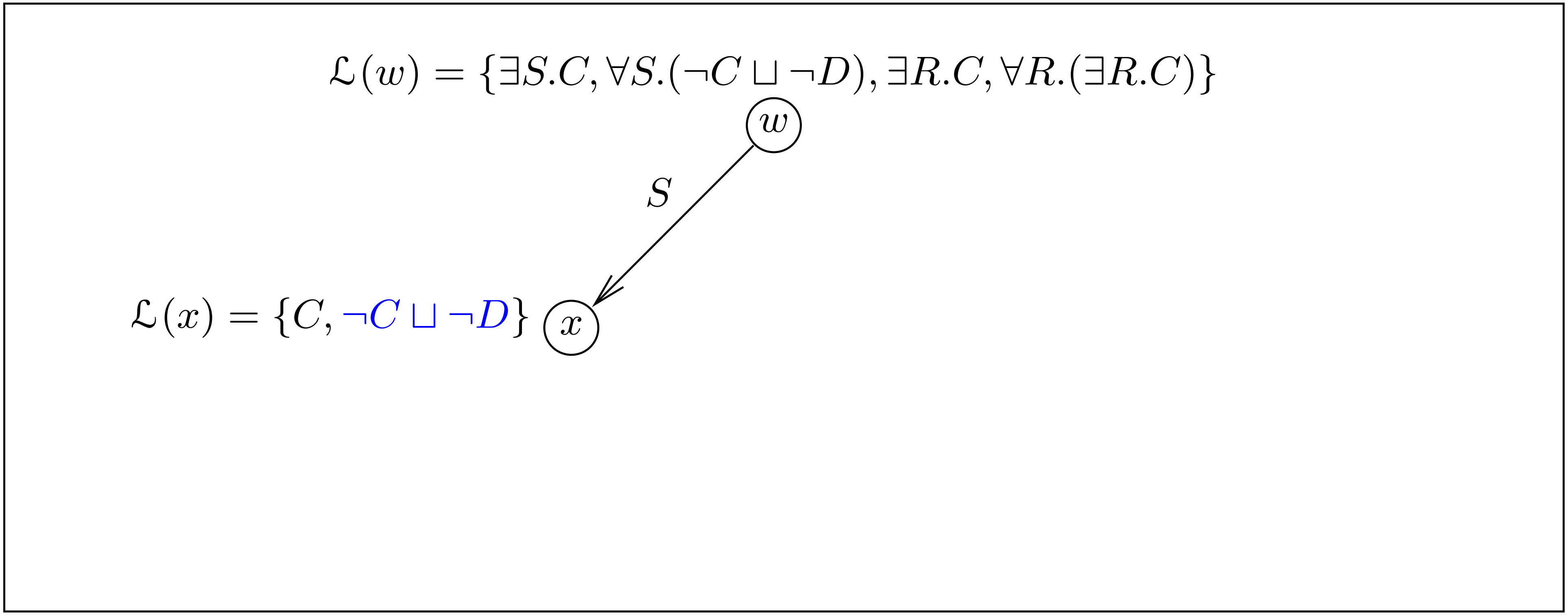
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



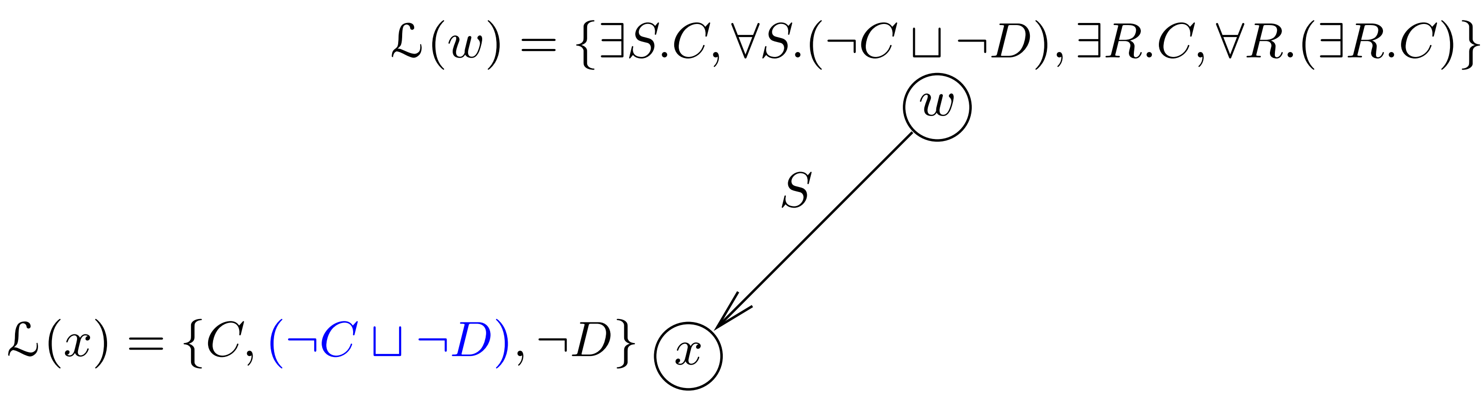
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



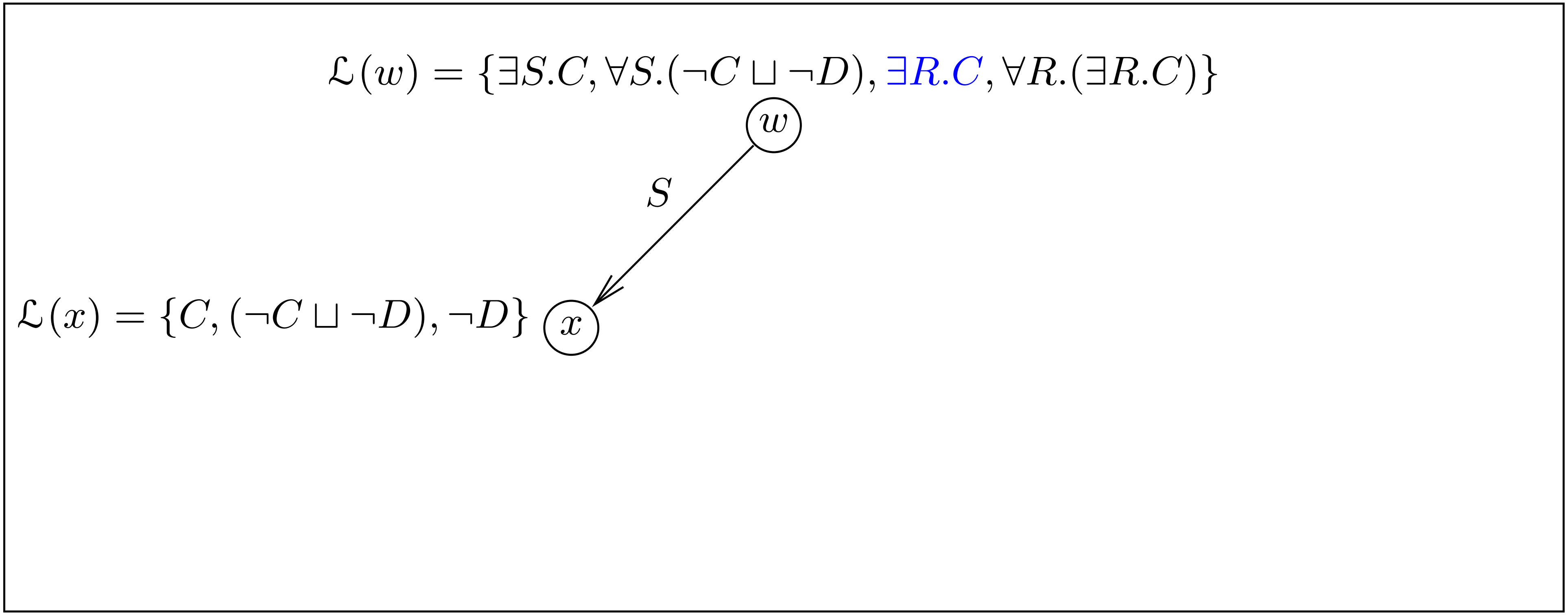
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

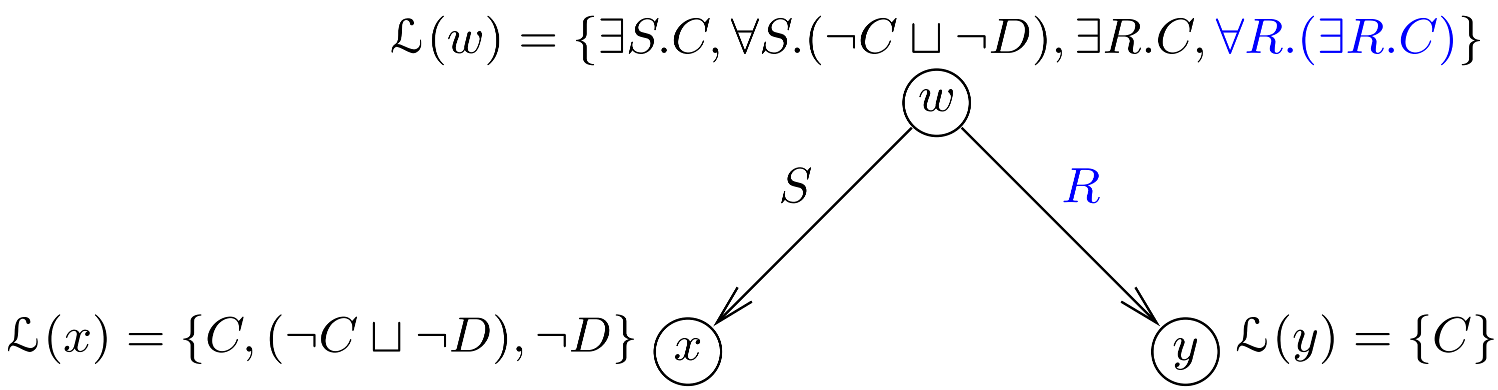
$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$

$$\begin{tikzcd} & w \arrow[dl, "S"'] \arrow[dr, "R"] & \\ x & & y \end{tikzcd}$$

$$\mathcal{L}(x) = \{C, (\neg C \sqcup \neg D), \neg D\} \qquad \mathcal{L}(y) = \{C\}$$

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$

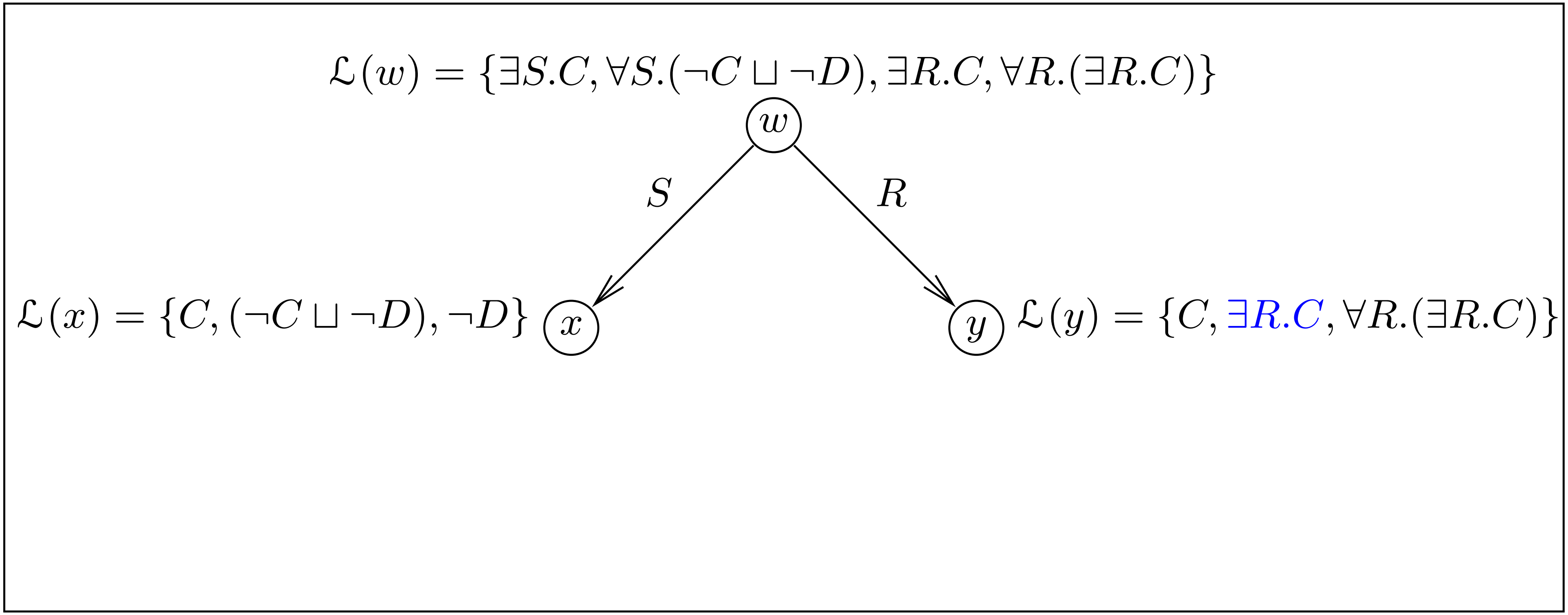
$$\begin{tikzcd} & w \arrow[dl, "S"'] \arrow[dr, "R"] & \\ x & & y \end{tikzcd}$$

$$\mathcal{L}(x) = \{C, (\neg C \sqcup \neg D), \neg D\}$$

$$\mathcal{L}(y) = \{C, \exists R.C, \forall R.(\exists R.C)\}$$

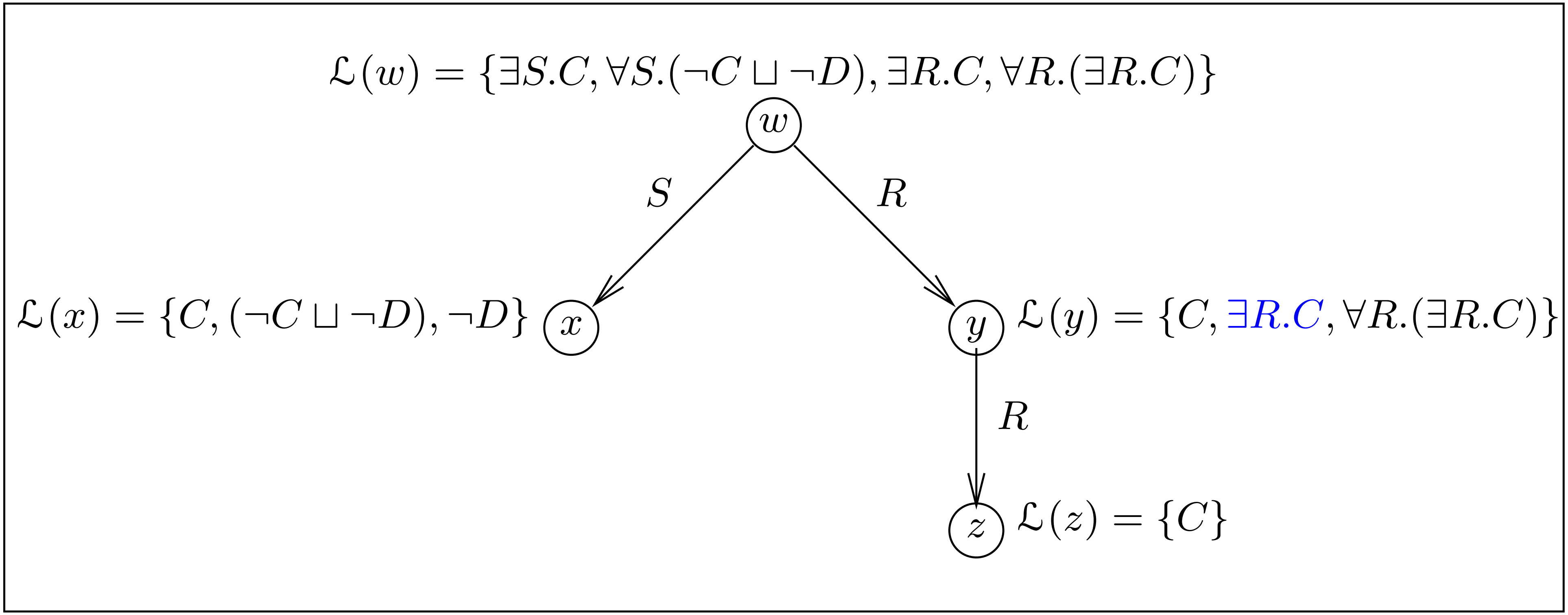
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



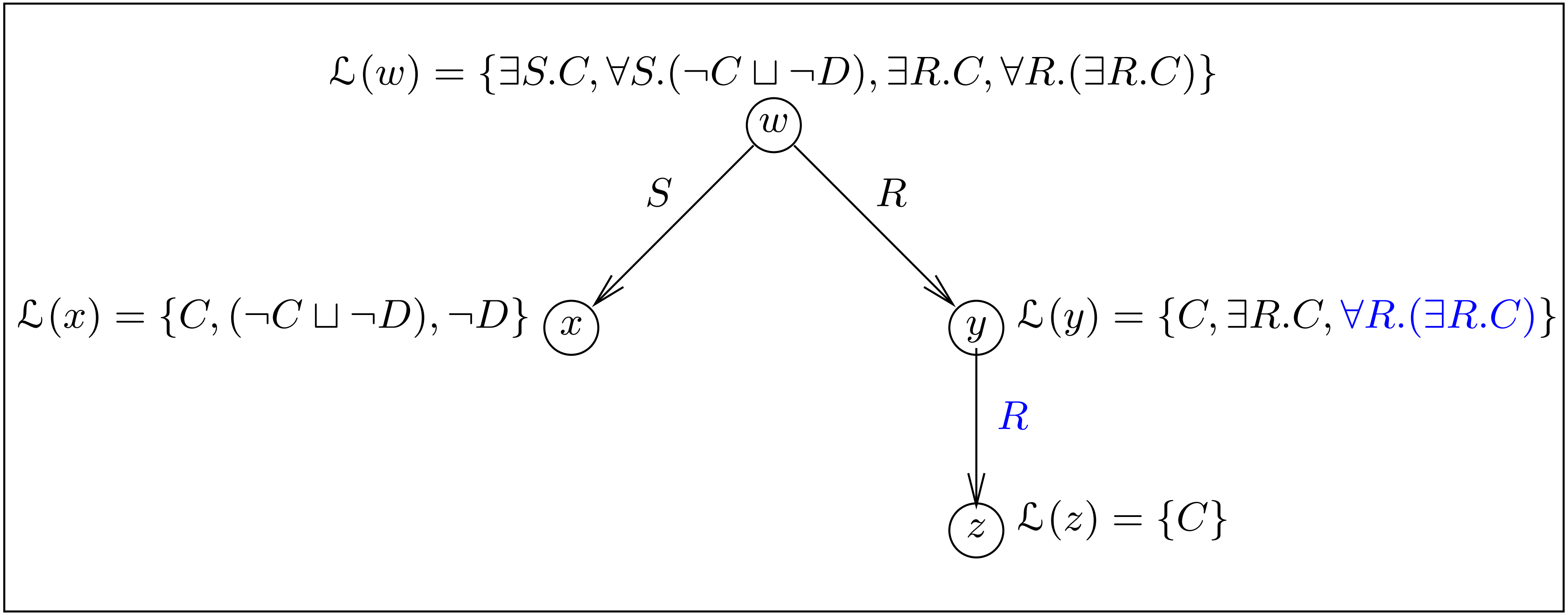
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



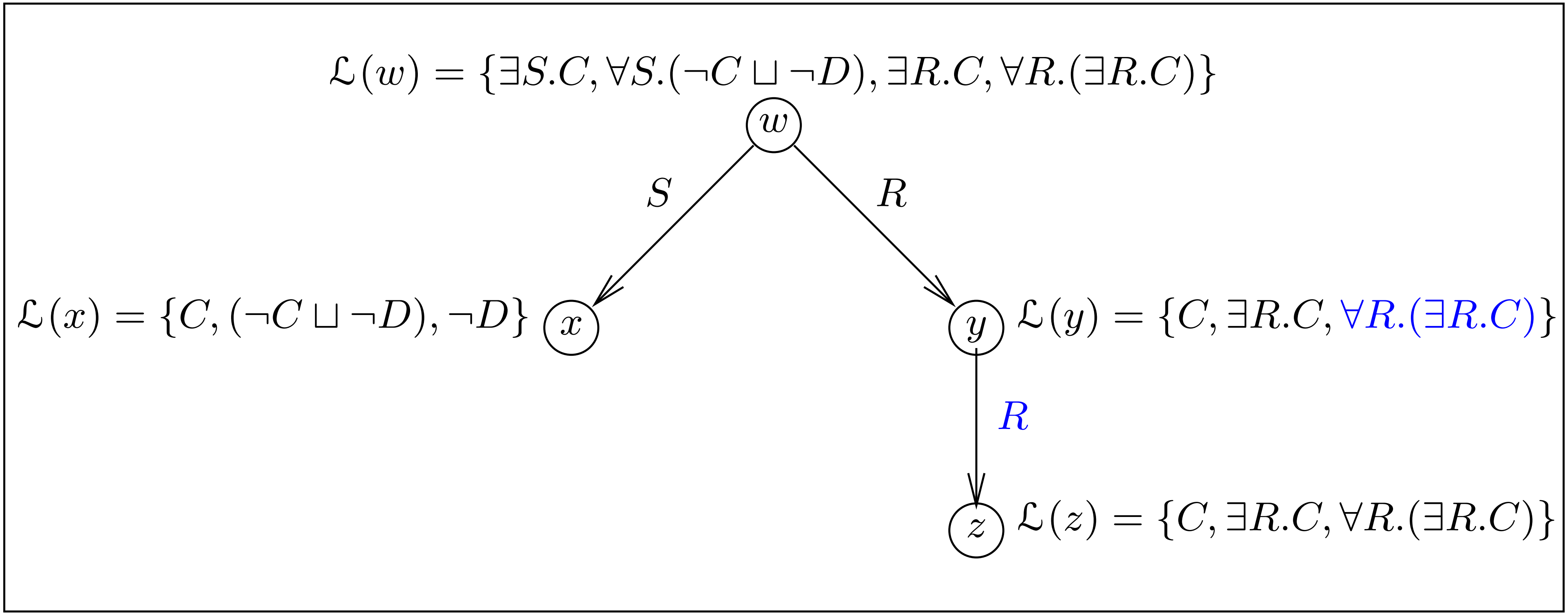
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



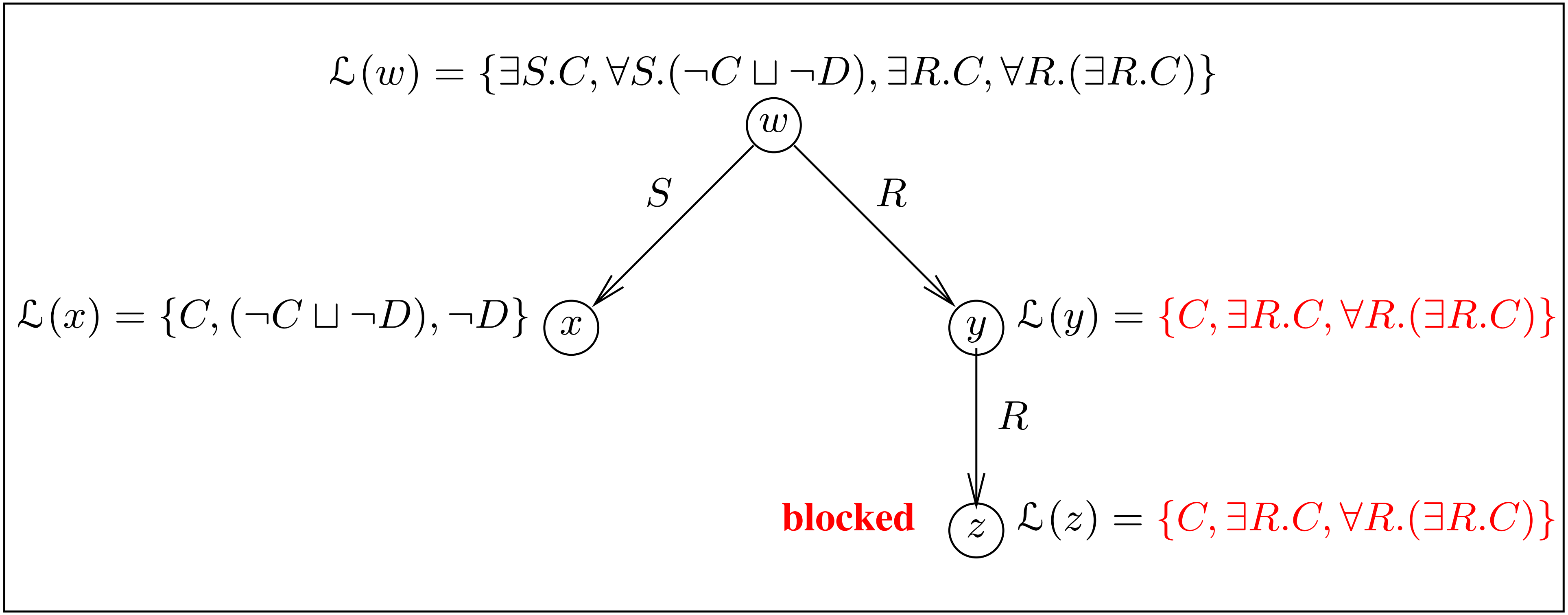
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



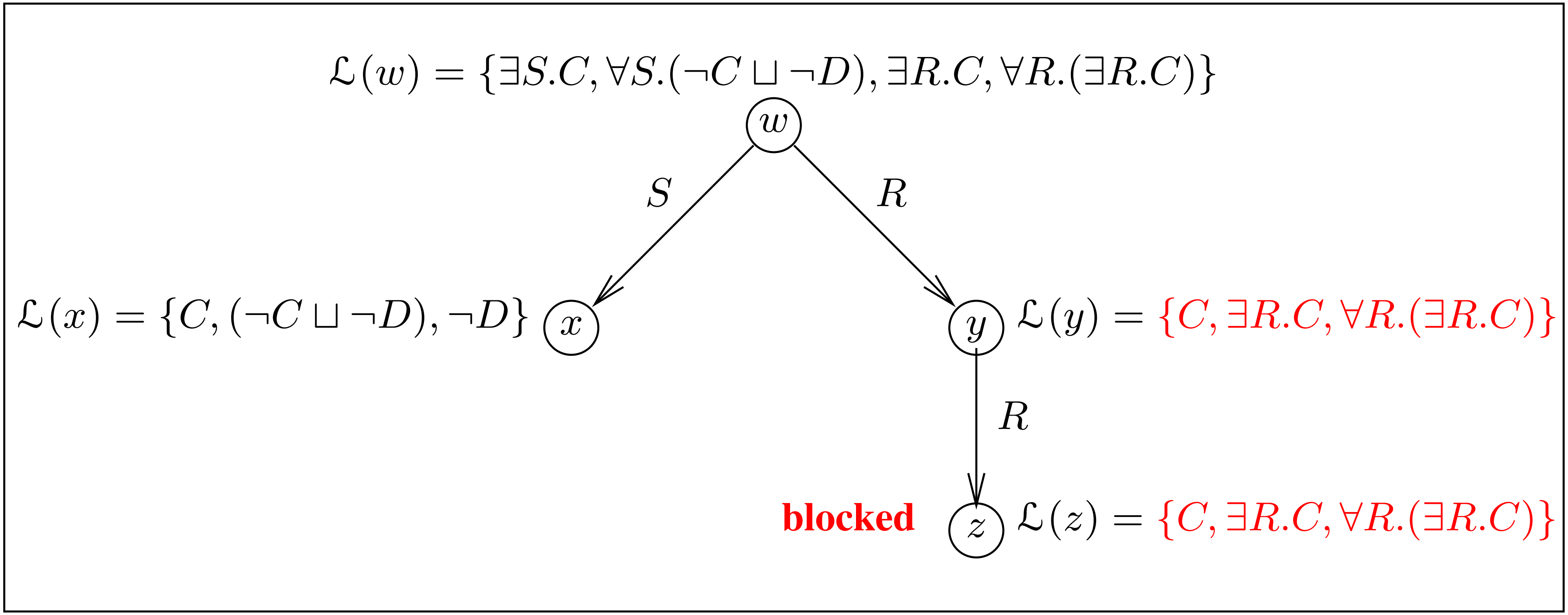
Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Algorithm — Example

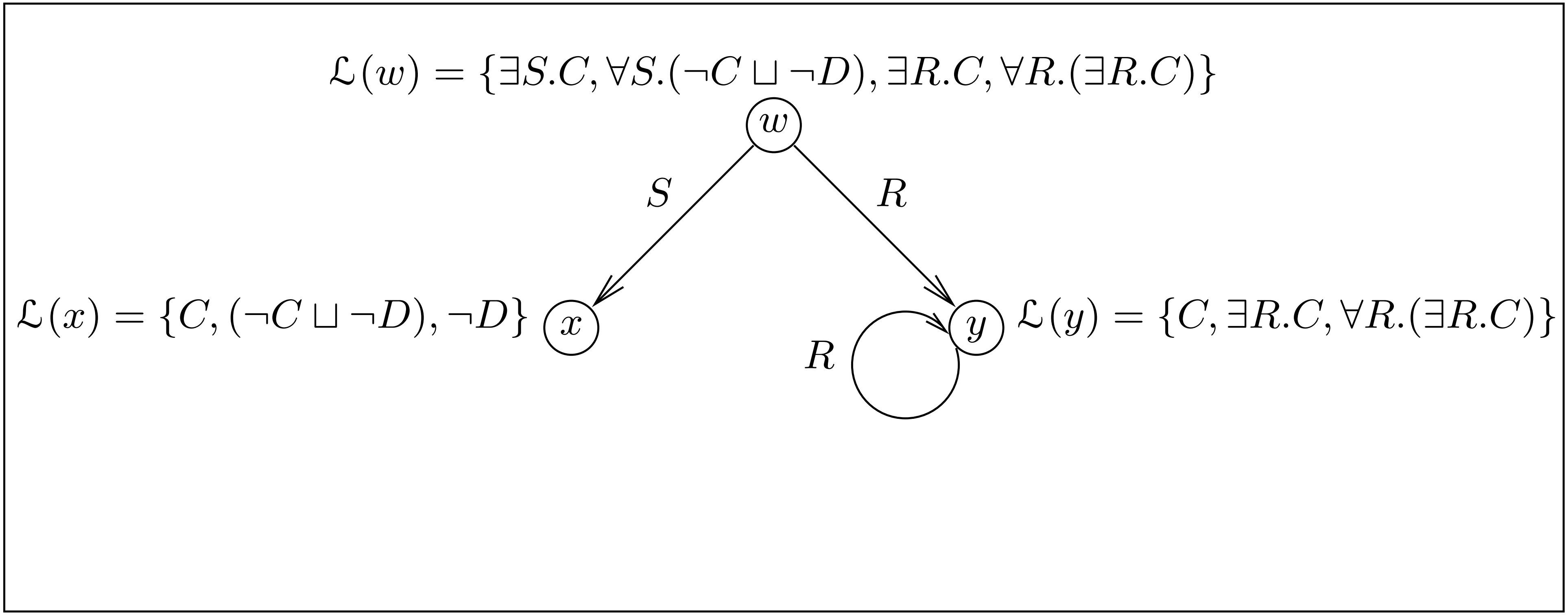
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Concept is **satisfiable**: $\mathbf{T}$ corresponds to **model**

Tableaux Algorithm — Example

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Concept is **satisfiable**: $\mathbf{T}$ corresponds to **model**

Representación de Conocimientos: Lógica formal y lógica descriptiva

Oscar Corcho García
(basado en transparencias de Asunción Gómez Pérez, Mariano Fernández López, Sean Bechhofer e Ian Horrocks)
ocorcho@fi.upm.es
Despacho 2107

Departamento de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain