

Lógica dinámica – Dynamic logic

Ma. Laura Cobo

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Argentina

Departamento de Ciencias e Ingeniería de la Computación – Universidad Nacional del Sur, Argentina

Hasta ahora ...

Utilizamos **JML** para la especificación de programas Java

Introducimos una **variante de la lógica de primer orden** que nos sirvió de marco para describir y razonar sobre estructuras de datos, relaciones entre objetos, valores de variables perono así sobre estados de programas

La idea ahora es ... extender la lógica y el cálculo para describir y razonar sobre el comportamiento de los programas (se requiere considerar no solo uno sino varios estados del programa)

Motivación

Se desea contar con una lógica-cálculo que permite expresar-
probar propiedades

Veamos un ejemplo:

Propiedad: **if** $a \neq \text{null}$ **then** **doubleContent** termina normalmente
and todos los elementos tienen el doble de su valor anterior.

método:

```
public doubleContent (int [] a) {  
    int i = 0;  
    while (i < a.length) {  
        a[i] = a[i]*2;  
        i++;  
    }  
}
```

Motivación

Propiedad: **if** $a \neq \text{null}$ **then** `doubleContent` termina normalmente **and** todos los elementos tienen el doble de su valor anterior.

En **lógica dinámica (DL)** la propiedad se vería de esta manera:

```
a ≠ null
^ a ≠ b
^ ∀ int i; ((0 ≤ i ^ i < a.length) → a[i] = b[i])
→ { doubleContent(a); }
   ∀ int i; ((0 ≤ i ^ i < a.length) → a[i] = 2 * b[i])
```

**Se combina FOL con programas.
DL extiende FOL**

Motivación

Lógica dinámica = extensión de FOL con

- Interpretaciones dinámicas
- Programas que describen cambios de estados

Repasemos los conceptos de FOL ...

FOL: aspectos semánticos

Es necesario refinar varias nociones. En FOL un sistema de tipos estaba definido como:

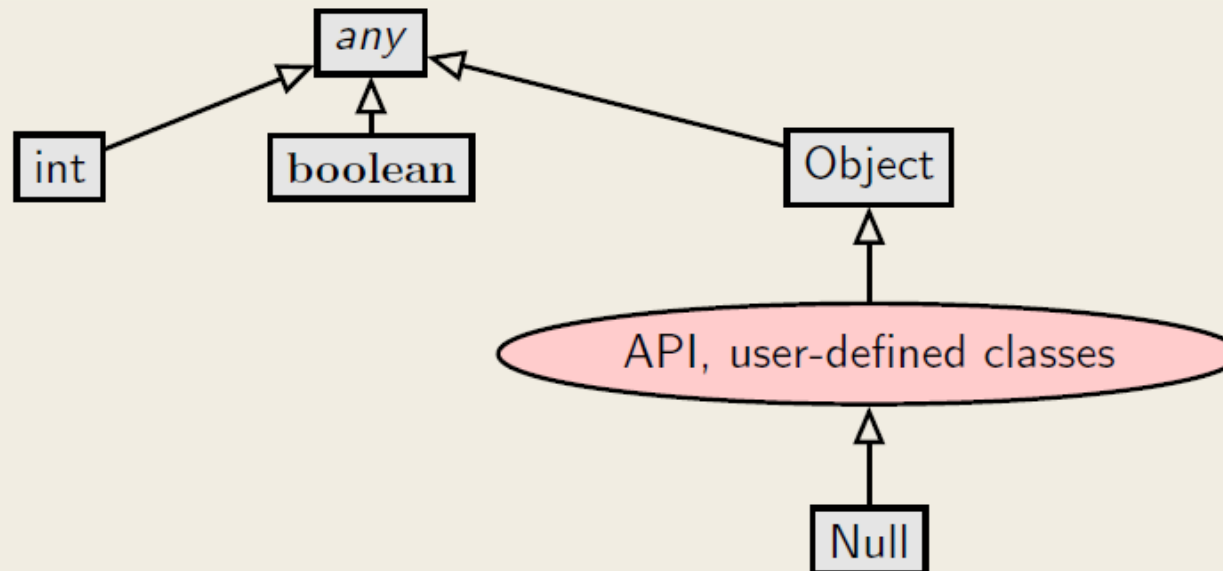
Definition (Type Hierarchy)

- ▶ T_{Σ} is set of **types**
- ▶ Given **subtype** relation ' \sqsubseteq ', with top element '*any*'
- ▶ $\tau \sqsubseteq \text{any}$ for all $\tau \in T_{\Sigma}$

El sistema minimal considera un único tipo: **any**. Todos los símbolos de signatura tienen el mismo tipo

Jerarquía de tipos en Java

Signature based on Java's type hierarchy (simplified)



FOL: clases y atributos

Person
<code>int age</code> <code>int id</code>
<code>int setAge(int newAge)</code> <code>int getId()</code>

- El dominio de todas los objetos Person: $\mathcal{D}^{\text{Person}}$
- Cada objeto $o \in \mathcal{D}^{\text{Person}}$ tiene asociado un valor edad (age)
- $\mathcal{I}(\text{age})$ es el mapeo desde Persona a int.
- Para cada clase C con atributo de tipo a: FSym declara una función $a(C)$ de ese tipo.

Field Access

Signature FSym: `int age(Person);` `Person p;`

Java/JML expression `p.age >= 0`

Typed FOL `age(p) >= 0`

KeY postfix notation for FOL `p.age >= 0`

Navigation expressions in KeY look exactly as in JAVA/JML

FOL: hacia una vista dinámica

En FOL solo se pueden expresar propiedades estáticas, por ejemplo:

- Valores de atributos en un cierto rango
- Propiedades (invariantes) de una subclase que implican propiedades de la superclase
- Etc.

La idea es poder expresar propiedades funcionales del programa

Como por ejemplo:

If el método `setAge` es invocado sobre un objeto `o` de tipo `Person`

And el argumento del método, de nombre `newAge`, es positivo

Entonces el atributo `age` del objeto `o`, tiene el mismo valor que `newAge`

FOL: hacia una vista dinámica

La lógica debe permitirnos:

- Relacionar diferentes estados del programa (**antes** y **después** de la ejecución en una sola fórmula)
- Las variables/atributos del programa representados mediante símbolos constantes/funciones dependen del estado del programa.

La lógica dinámica es apropiada para cubrir estos requerimientos

Lógica dinámica

La lógica dinámica (para Java) es:

lógica de primer orden tipada

- + programas p
- + modalidades $\langle p \rangle \phi$, $[p] \phi$ (p es un programa, ϕ es una fórmula DL)
- +

$$i > 5 \rightarrow [i = i + 10;] i > 15$$

Si una variable del programa, i , es mayor que 5 entonces luego de ejecutar $i = i + 10$, i es mayor que 15

Lógica dinámica

$$i > 5 \rightarrow [i = i + 10;] i > 15$$

La variable i hace referencia a los valores de la misma antes y después de la ejecución del programa.

Las variables del programa, como i , son constantes simbólicas que dependen del estado. El valor de los símbolos dependientes del estado cambian durante la ejecución del programa.

A este tipo de símbolos se los suele llamar: flexibles, estado-dependientes, no-rígidos

Lógica dinámica: signatura

Una signatura en DL está definida de la misma manera que para FOL pero agrega símbolos flexibles

Signature

A first-order signature Σ consists of

- ▶ a set T_Σ of types
- ▶ a set F_Σ of function symbols
- ▶ a set P_Σ of predicate symbols
- ▶ a typing α_Σ

- **Rígido:** son los símbolos que tienen la misma interpretación en todos los estados del programa.

Variables de primer orden – funciones y predicados built-in

- **Flexible:** son los símbolos cuya interpretación depende del estado de la ejecución.

Variables del programa y atributos

Lógica dinámica: signature

Una signature en DL está definida de la misma manera que para FOL pero agrega símbolos flexibles

- **Rígido:** son los símbolos que tienen la misma interpretación en todos los estados del programa.

Variables de primer orden – funciones y predicados built-in

- **Flexible:** son los símbolos cuya interpretación depende del estado de la ejecución.

Variables del programa y atributos

Las variables de primer orden son aquellas que están declaradas localmente en los cuantificadores (no pueden ser variables del programa)

Lógica dinámica: signatura – formato del archivo KeY

Las secciones vacías pueden no incluirse

```
\sorts {  
    // only additional sorts (predefined:  int/boolean/any)  
}  
\functions {  
    // only additional rigid functions  
    // (arithmetic functions like +,- etc.  predefined)  
}  
\predicates { /* same as for functions */ }  
  
\programVariables { // flexible functions  
    int i, j;  
    boolean b;  
}
```

Lógica dinámica: Términos

Son como los definidos para FOL pero pueden contener tanto símbolos flexibles como rígidos

Terms

A first-order term of type $\tau \in T_\Sigma$

- ▶ is either a variable of type τ , or
- ▶ has the form $f(t_1, \dots, t_n)$,
where $f \in F_\Sigma$ has result type τ , and each t_i is term of the correct type, following the typing α_Σ of f .

Lógica dinámica: programas

Un programa en este contexto es cualquier secuencia legal de sentencias Java.

```
Signature for FSymf: int r; int i; int n;  
Signature for FSymr: int 0; int +(int,int); int -(int,int);  
Signature for PSymr: <(int,int);  
  
i=0;  
r=0;  
while (i<n) {  
    i=i+1;  
    r=r+i;  
}  
r=r+r-n;
```

¿cuál es el valor computado en r?

Lógica dinámica: relacionando estados del prog.

DL agrega dos operadores nuevos.
Estos operadores son modales

- $\langle p \rangle \phi$ (diamante) p es un programa
- $[p] \phi$ (cuadrado) ϕ es otra formula DL

Significado intuitivo

- $\langle p \rangle \phi$: p termina **y** la fórmula ϕ se verifica en el estado final (correctitud total)
- $[p] \phi$: **SI** p termina **entonces** la fórmula ϕ se verifica en el estado final (correctitud parcial)

Los programas Java son deterministas

Lógica dinámica: fórmula

Formulas

- ▶ each atomic formula is a formula
- ▶ with ϕ and ψ formulas, x a variable, and τ a type, the following are also formulas:
 - ▶ $\neg\phi$ ("not ϕ ")
 - ▶ $\phi \wedge \psi$ (" ϕ and ψ ")
 - ▶ $\phi \vee \psi$ (" ϕ or ψ ")
 - ▶ $\phi \rightarrow \psi$ (" ϕ implies ψ ")
 - ▶ $\phi \leftrightarrow \psi$ (" ϕ is equivalent to ψ ")
 - ▶ $\forall \tau x; \phi$ ("for all x of type τ holds ϕ ")
 - ▶ $\exists \tau x; \phi$ ("there exists an x of type τ such that ϕ ")

Definition (Dynamic Logic Formulas (DL Formulas))

- ▶ Each FOL formula is a DL formula
- ▶ If p is a program and ϕ a DL formula then $\left\{ \begin{array}{l} \langle p \rangle \phi \\ [p] \phi \end{array} \right\}$ is a DL formula
- ▶ DL formulas closed under FOL quantifiers and connectives

Lógica dinámica: fórmulas

Definition (Dynamic Logic Formulas (DL Formulas))

- ▶ Each FOL formula is a DL formula
- ▶ If p is a program and ϕ a DL formula then $\left\{ \begin{array}{l} \langle p \rangle \phi \\ [p] \phi \end{array} \right\}$ is a DL formula
- ▶ DL formulas closed under FOL quantifiers and connectives

Observaciones

- Las variables del programa son constantes flexibles, por lo tanto no pueden estar ligadas en cuantificadores.
- Las variables del programa no necesitan ser declaradas o inicializadas en el programa.
- Los programas no contienen variables lógicas
- Los operadores modales pueden anidarse en forma arbitraria.

Lógica dinámica: estado

Program states as first-order states

From now, consider program state s as **first-order state** $(\mathcal{D}, \delta, \mathcal{I})$

- ▶ Only interpretation \mathcal{I} of flexible symbols in FSym_f can change
⇒ only record values of $f \in \text{FSym}_f$
- ▶ *States* is set of all states s

El estado sigue representado por el dominio y la interpretación

Lógica dinámica: semántica

Definition (Kripke Structure)

Kripke structure or Labelled transition system $K = (States, \rho)$

- ▶ **State** (=first-order model) $s = (\mathcal{D}, \delta, \mathcal{I}) \in States$
- ▶ **Transition relation** $\rho : Program \rightarrow (States \rightarrow States)$

$$\rho(p)(s1) = s2$$

iff.

program p executed in state $s1$ terminates **and** its final state is $s2$,
otherwise undefined.

La relación de transición define la semántica de los programas.

La relación de transición del programa a partir de s puede no estar definida, esto indica que el programa puede no terminar comenzando desde el estado s .

La relación es un **función**, ya que los programas son deterministas.

Dynamic Logic Calculus

Ma. Laura Cobo

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Argentina

Departamento de Ciencias e Ingeniería de la Computación – Universidad Nacional del Sur, Argentina

Vinculación a la lógica de Hoare

Las fórmulas de correctitud tienen la forma:

$$\{\text{pre}\} P \{\text{post}\}$$

Siendo pre y post fórmulas de primer orden

Al utilizar lógica dinámica este tipo de formulas de correctitud se exprepresan como:

$$\text{Pre} \rightarrow [P] \text{Post}$$

Siendo Pre y Post cualquier fórmula de lógica dinámica

**La clave ahora está en cómo probar que una fórmula es válida,
es decir, se satisface en todos los estados**

Ejecución simbólica

La ejecución simbólica sigue el flujo de control natural al momento de analizar el programa.

Algunas variables tienen valor desconocido: luego se utiliza una **representación simbólica del estado**.

- La regla simbólica ejecuta la primer sentencia (o sentencia activa)
- La aplicación repetida de la regla corresponde a la **ejecución simbólica del programa**.

Ejecución simbólica

Symbolic execution of conditional

$$\text{if} \frac{\Gamma, b \doteq \text{true} \Rightarrow \langle p; \text{rest} \rangle \phi, \Delta \quad \Gamma, b \doteq \text{false} \Rightarrow \langle q; \text{rest} \rangle \phi, \Delta}{\Gamma \Rightarrow \langle \text{if } (b) \{ p \} \text{ else } \{ q \} ; \text{rest} \rangle \phi, \Delta}$$

Symbolic execution of loops: unwind

$$\text{unwindLoop} \frac{\Gamma \Rightarrow \langle \text{if } (b) \{ p; \text{while } (b) p \}; \text{rest} \rangle \phi, \Delta}{\Gamma \Rightarrow \langle \text{while } (b) \{ p \}; \text{rest} \rangle \phi, \Delta}$$

Updates

Se necesita una notación para los cambios de estados simbólicos

- La ejecución simbólica debe “caminar” por el programa siguiendo el flujo de ejecución natural.
- Requiere una representación sucinta de los cambios de estado efectuados por el programa en una ejecución simbólica ramificada
- Se desea simplificar los efectos de la ejecución temprana de programas.
- Se desea aplicar los efectos tarde (en la condiciones de ramificación y post-condiciones)

Se utiliza una notación dedicada para cambios simple de estados: updates

Updates: estados explícitos

Definition (Syntax of Updates, Updated Terms/Formulas)

If v is program variable, t FOL term type-compatible with v ,
 t' any FOL term, and ϕ any DL formula, then

- ▶ $v := t$ is an update
- ▶ $\{v := t\}t'$ is DL term
- ▶ $\{v := t\}\phi$ is DL formula

Definition (Semantics of Updates)

State s interprets flexible symbols f with $\mathcal{I}_s(f)$

β variable assignment for logical variables in t , ρ transition relation:

$\rho(\{v := t\})(s, \beta) = s'$ where s' identical to s except $\mathcal{I}_{s'}(v) = \text{val}_{s, \beta}(t)$

Updates

Detalles sobre los updates $\{v := t\}$

- La semántica del update es casi idéntica a la de las asignaciones
- El valor del update depende también de las variables lógicas en t , es decir β
- Los updates no son asignaciones: el lado derecho es un término FOL.
 - ✓ $\{x := n\} \phi$ no puede transformarse en una asignación (n variable lógica)
 - ✓ $\{x = i++\} \phi$ no puede transformarse directamente en update.
- Los updates no son ecuaciones: cambian los valores de los términos flexibles

Updates

Symbolic execution of assignment using updates

$$\text{assign} \frac{\Gamma \Rightarrow \{x := t\} \langle \text{rest} \rangle \phi, \Delta}{\Gamma \Rightarrow \langle x = t; \text{rest} \rangle \phi, \Delta}$$

- Es simple
- Funciona bien siempre y cuando el lado derecho no tenga efectos colaterales (esto se garantiza en DL simple)
- Se requieren casos especiales para situaciones como $x = t_1 + t_2$

Updates: otros usos

**No esta permitido cuantificar sobre variables del programa
pero Se puede cuantificar sobre otro valor y asignarlo a una
variable del programa**

$$\forall \tau \text{ } i_0; \{i := i_0\} \langle \dots i \dots \rangle \phi$$