Processadors de Llenguatges

Pràctica III: Anàlisi semàntica i gestió de la taula de símbols

Curs 2023/24

Objectius

La pràctica que proposem té com objectiu integrar les fases d'anàlisi d'un procés de traducció amb la gestió de la taula de símbols. El treball consistirà en extendre el llenguatge de les fòrmules ben formadas de CP1, especificat a la pràctica 2, amb l'anàlisi semàntica.

Exercici 1 Anàlisi semàntica de les fòrmules de CP1 (4 punts)

A la pràctica anterior vam implementar un analitzador sintàctic per a fòrmules de CP1. Ara hem d'extendre la implementació amb l'analitzador semàntic de manera que reconegui fòrmules de CP1 ben formades, però sense variables lliures i considerant definicions estàtiques de les funcions i dels predicats dins de cada àmbit definit per un quantificador. És a dir, totes les variables han d'estar quantificades universal o existencialment (variables lligades) i, la primera definició d'una variable, funció o d'un predicat dins de l'àmbit introduït per un quantificador, estableix la definició amb el corresponent perfil (funcions i predicats) fins tancar l'àmbit.

Per facilitar la lectura, a nivell lèxic, considerarem els operadors lògics de conjunció i disjunció especificats amb les paraules reservades `and` i `or`.

A mode de recordatori: La precedència de major (1) a menor (5) prioritat dels operadors és la següent:

1. $\neg$ (associatiu per la dreta)
2. $\forall$ i $\exists$ (associatius per la dreta)
3. $\wedge$ (associatiu per l'esquerra)
4. $\vee$ (associatiu per l'esquerra)
5. $\rightarrow$ i $\leftrightarrow$ (associatiu per l'esquerra)

D'altra banda, els parèntesis permeten modificar la prioritat associada amb els operadors.

Finalment, comentar que la sortida de l'analitzador serà els possibles missatges d'error i la indicació de si la fòrmula és o no correcta.

Per exemple, les fòrmules següents són lèxica, sintàctica i semànticament correctes:

```
// Correctes a nivell lèxic sintàctic i semàntic

forall x1 forall x2 (P1(x1, x2) -> P2(x2));

exists x1  (P1(x1) -> forall x2 P2(x2));

P1(a1,a2) and forall x1 forall x2 (P1(x1, x2) -> P2(x2));

forall x1 forall x2 (P1(x1, x2) -> P2(x2)) and P1(a1,a2);

forall x1 forall x2 (P1(x1, x2) -> P2(x2)) and P1(b1);

forall x1 exists x2 (P1(f1(x1, x2)) -> P2(f2(x2))) and P1(f1(a1),f2(b1,b2),f3(c1));
```

Una possible sortida per als exemples anteriors és la següent (únicament hem emès missatge per a les variables, però es creen entrades per a tots els elements):

```
Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 3

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 5

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 7
```

```
Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 9

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 11

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 13
```

En canvi, les fòrmules següents són semànticament in correctes:

```
// Incorrectes semàntic

// Variable x2 no definida

forall x1 (P1(x1, x2)-> P2(x2));

// Redefinició de P1

P1(a1) and P1(b1,b2);

// Redefinició de P1 dins de l'ambit 2n forall

forall x1 forall x2 (P1(x1, x2)-> P1(x2));

// Redefinició de P1
```

```
P1(a1) and forall x1 forall x2 (P1(x1, x2) -> P2(x2));

// Redefinició de f1 dins de l'ambit 2n forall

forall x1 forall x2 (P1(f1(x1, x2))-> P2(f1(x2)));

// x2 variable lliure a l'àmbit 1r forall

(forall x1 (P1(x1,x2)<->(exists x2 P2(x2) and P3(x1,x2) or !P2(f1(x1,x2)))));

// x2 variable lliure a l'àmbit 1r forall

(forall x1 (P1(x1,f2(f1(a1))))<->(exists x2 P2(x2) and P3(x1,x2)
or !P2(f1(x1,x2))));

// Redefinició de variable x2

(forall x1 exists x2 (P1(x1,x2)<->(forall x2 P2(x2) and P3(x1,x2)
or !P2(f1(x1,x2)))));
```

Per exemple, una possible sortida per als exemples anteriors és la següent:

```
Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
ERROR semàntic. Variable LLIURE: x2  Línea 5
ERROR Fòrmula INCORRECTA Línea 5

Empila àmbit ID 0
ERROR Semàntic Línea 9
Redefinició Predicat P1 amb aritat 1 a aritat 2
ERROR Fòrmula INCORRECTA Línea 9

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
ERROR Semàntic Línea 13
Redefinició Predicat P1 amb aritat 2 a aritat 1
ERROR Fòrmula INCORRECTA Línea 13

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
```

```
ERROR Semàntic Línea 17
Redefinició Predicat P1 amb aritat 1 a aritat 2
ERROR Fòrmula INCORRECTA Línea 17

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
ERROR Semàntic Línea 21
Redefinició Funcio f1 amb aritat 2 a nova aritat 1
ERROR Fòrmula INCORRECTA Línea 21

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
ERROR semàntic. Variable LLIURE: x2  Línea 25
ERROR Fòrmula INCORRECTA Línea 25

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Desempila àmbit ID 1
Empila àmbit ID 1
Crea variable x2
Desempila àmbit ID 1
ERROR semàntic. Variable LLIURE: x1  Línea 29
ERROR Fòrmula INCORRECTA Línea 29

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1
Empila àmbit ID 2
Crea variable x2
Empila àmbit ID 3
ERROR Semàntic Línea 33
Redefinició Variable Quantificada x2
ERROR Fòrmula INCORRECTA Línea 33
```

Exercici 2 Definició i gestió de tipus a les fòrmules de CP1 (4 punts)
Extendre l'exercici anterior amb un sistema de definició de tipus estàtic per a les fòrmules de CP1.

Característiques del sistema de tipus proposat:

- Considereu 4 tipus bàsics, o dominis. Per exemple, podem considerar els tipus `bool`, `char`, `int` i `real`.

- Heu de definir un mecanisme sintàctic per definir el tipus de cada component del llenguatge. En aquest cas: constants, variables, funcions i predicats.

  Per exemple, podem considerar una sintaxi basada en l'ús dels 2 punts de la forma següent:

```
( forall x1:int exists x2:real ( P1(f1(x1, x2):real)  ->
P2(f2(x2):int) and P2(x1) ) or
P1(f1(a1:int):int, f2(a1, b2:char):char , f1(c1:int)) ) ;
```

  És a dir, cada definició va succeïda del tipus.

- Els símbols de predicats, només tenen tipus per als termes, ja que sempre s'avaluen a valors Boolean (codomini). A més, els predicats no són termes, que seria el cas de les lògiques d'ordres superiors.

- Un cop definit un component del llenguatge dins d'un àmbit, el seu tipus romandrà estàtic fins tancar l'àmbit. Per tant, l'especificació de tipus només cal realitzar-la a la primera aparició del component. Per tant, quan un component ja està definit dins d'un àmbit, no cal especificar el tipus.

- En referència al sistema de tipus, podeu considerar un sistema de tipus estricte, basat en un casament estricte del tipus dels paràmetres, o considerar un sistema de tipus més laxe on es permet la conversió de tipus més específics a tipus més generals.

- Considerant un **sistema de tipus estricte** la sortida per a la fòrmula antrior pot ser:

```
Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta
```

Per exemple, considerant un **sistema de tipus estricte**, les fòrmules següents són correctes:

```
// Correctes a nivell lèxic sintàctic i semàntic amb casament de tipus

forall x1:int forall x2:real (P1(x1, x2) -> P2(x2));

exists x1:char  (P1(x1) -> forall x2:char P2(x2));

forall x1:int forall x2:char (P1(x1, x2) -> P2(x2)) and P1(a1:int,a2:char);
```

```
forall x1:int forall x2:char (P1(x1, x2) -> P2(x2)) and P1(b1:real);

forall x1:int exists x2:real (P1(f1(x1, x2):int) -> P2(f2(x2):char))
and P1(f1(a1:int):int);
```

Una possible sortida per als exemples anteriors és la següent (considerant un sistema de tipus estricte i generant missatges únicament per a les variables, però es creen entrades amb la informació de tipus per a tots els elements):

```
Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 3

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 2
Empila àmbit ID 2
Crea variable x2   Tipus 2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 5

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 7

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
```

```
Fòrmula CP1 Correcta Línia 9

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
Fòrmula CP1 Correcta Línia 11
```

En canvi, les fòrmules següents són semànticament incorrectes:

```
// Fòrmules Incorrectes a nivell semàntic

// Variable x2 no definida

forall x1:int (P1(x1, x2)-> P2(x2));

// Redefinició de P1 dins de l'ambit 2n forall

forall x1:int forall x2:real (P1(x1, x2)-> P1(x2));

// x2 variable lliure a l'àmbit 1r forall

(forall x1:int (P1(x1,x2)<->(exists x2:int P2(x2))));

// Redefinició P1(int) a P1(real)

(forall x1:int (P1(x1)<->(exists x2:real P1(x2) and P2(x2))));

// Redefinició P1(int,real) a P1(int)

forall x1:int exists x2:real (P1(x1, x2)<-> P1(x1));

// Redefinició P1(int,real) a P1(real, int)

forall x1:int exists x2:real (P1(x1, x2) and P1(x2, x1));

// Redefinició de constant a1 de int a real

forall x1:real (P1(x1, a1:int) -> P2(a1:real));

// Constant a2 sense tipus

forall x1:real P1(x1, a1:int) -> P2(a2);
```

```
// Constant a1 a P2 sense tipus (int únicament a l'àmbit forall)

forall x1:real P1(x1, a1:int) -> P2(a1);

// Redefinició de f1 dins de l'ambit 2n forall

forall x1:int forall x2:int (P1(f1(x1, x2):int)-> P2(f1(x2)));

// f2 definida sense tipus

forall x1:int (P1(x1,f2(f1(a1:int):char)));

// Redefinició de funció f1

forall x1:int exists x2:char (P1(f1(x1,x2):int) or !P2(f1(x2,x1)));
```

Una possible sortida per als exemples anteriors és la següent (considerant un sistema de tipus estricte):

```
Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
ERROR semàntic Línea 5
Variable LLIURE: x2
 ERROR Fòrmula INCORRECTA Línea 5

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
ERROR Semàntic Línea 9
Redefinició Predicat P1 amb aritat 2 a aritat 1
ERROR Fòrmula INCORRECTA Línea 9

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
ERROR semàntic Línea 13
Variable LLIURE: x2
 ERROR Fòrmula INCORRECTA Línea 13

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
ERROR Semàntic Línea 17
```

```
Redefinició Predicat P1 amb aritat 1. Terme 0 definit com 3 i redefinit a  4
ERROR Fòrmula INCORRECTA Línea 17

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
ERROR Semàntic Línea 21
Redefinició Predicat P1 amb aritat 2 a aritat 1
ERROR Fòrmula INCORRECTA Línea 21

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 4
ERROR Semàntic Línea 25
Redefinició Predicat P1 amb aritat 2. Terme 0 definit com 3 i redefinit a  4
ERROR Fòrmula INCORRECTA Línea 25

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 4
ERROR semàntic Línea 29
Redefinició de constant a1 de tipus 3 a nou tipus: 4
ERROR Fòrmula INCORRECTA Línea 29

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 4
Desempila àmbit ID 1
ERROR semàntic Línea 33
Constant sense tipus: a2
ERROR Fòrmula INCORRECTA Línea 33

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 4
Desempila àmbit ID 1
ERROR semàntic Línea 37
Constant sense tipus: a1
ERROR Fòrmula INCORRECTA Línea 37

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
```

```
Empila àmbit ID 2
Crea variable x2   Tipus 3
ERROR Semàntic Línea 41
Redefinició Funció f1 amb aritat 2 a aritat 1
ERROR Fòrmula INCORRECTA Línea 41

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
ERROR Semàntic Línea 45
Definició de funció  f2 sense tipus
ERROR Fòrmula INCORRECTA Línea 45

Empila àmbit ID 0
Empila àmbit ID 1
Crea variable x1   Tipus 3
Empila àmbit ID 2
Crea variable x2   Tipus 2
ERROR Semàntic Línea 49
Redefinició Funció f1 amb aritat 2. Terme 0 definit com 3 i redefinit a  2
ERROR Fòrmula INCORRECTA Línea 49
```

Exercici 3 Definició compacta de variables amb tipus a les fòrmules de CP1 (2 punts)
Extendre l'exercici anterior amb una notació compacta per a la definició de variables amb tipus a les fòrmules de CP1.

Fins ara la sintaxi que hem utilitzat és la següent:

```
(forall x1:int forall x2:int forall x3:int exists x4:char exists x5:char
(P1(x1, x2, x3) -> P2(x4, x5)));
```

En aquest exercici, l'objectiu és utilitzar una notació més compacta utilitzant el caràcter ',' de la forma:

```
(forall x1, x2, x3:int exists x4, x5:char  (P1(x1, x2, x3) -> P2(x4, x5)));
```

És a dir, permetre una llista de variables, totes del mateix tipus, associades amb un quantificador.

Una possible sortida per a l'exemple anterior és la següent:

```
Empila àmbit ID 0
Empila àmbit ID 1
3 Variables Quantificades de tipus 3:
Crea variable x1   Tipus 3
Crea variable x2   Tipus 3
```

```
Crea variable x3   Tipus 3
Empila àmbit ID 2
2 Variables Quantificades de tipus 2:
Crea variable x4   Tipus 2
Crea variable x5   Tipus 2
Desempila àmbit ID 2
Desempila àmbit ID 1
Desempila àmbit ID 0
```

Exercici 4 Exercici opcional 0,5 punts: Transformació d'expressions de CP1

Per obtenir la forma clausal d'una fòrmula de CP1 que únicament conté negacions, conjuncions i disjuncions, és necessari reduir al màxim l'àmbit d'aplicació dels símbolos de negació (els símbols de negació han d'actuar únicament sobre fòrmules atòmiques).

És a dir, si $F(x_1)$ denota una fòrmula de CP1 amb variables x_1, la reducció de l'àmbit d'aplicació de la negació es realitza de la forma següent:

$$\neg\forall x_1 F(x_1) \equiv \exists x_1 \neg F(x_1) \quad (1)$$
$$\neg\exists x_1 F(x_1) \equiv \forall x_1 \neg F(x_1) \quad (2)$$

On, si $F_1 \ldots F_n$ denoten fòrmules de CP1,

$$\neg(\neg F_1) \equiv F_1 \quad (3)$$
$$\neg(F_1 \vee \ldots \vee F_n) \equiv \neg(F_1) \wedge \ldots \wedge \neg(F_n) \quad (4)$$
$$\neg(F_1 \wedge \ldots \wedge F_n) \equiv \neg(F_1) \vee \ldots \vee \neg(F_n) \quad (5)$$
$$(6)$$

Es demana:

Implementar un traductor tal que donada una fòrmula de CP1 sense implicacions i dobles implicacions, generi com a sortida la fòrmula que s'obté com a resultat de reduir al màxim l'àmbit d'aplicació dels símbolos de negació (els símbols de negació hauran d'actuar únicament sobre fòrmules atòmiques o literals).

Per aquest exercici no cal que considereu la versió amb el sistema de tipus, és a dir, considereu el llenguatge de la pràctica anterior sense implicacions i dobles implicacions. D'altra banda, és recomanable construir l'arbre sintàctic a memòria per tal de realitzar el processat en 2 passos: (1) identificar l'estructura sintàctica de la fòrmula (identificar els quantificadors, les connectives i les fòrmules atòmiques) i (2) propagar les negacions dels nodes interns (quantificadors i connectives) cap a les fulles (fòrmules atòmiques).

Lliurament

La documentació a lliurar per a cada exercici de programació és la següent:

1. Especificació *lèxica*, *sintàctica*, semàntica i implementació de la taula de símbols.

2. Podeu utilitzar l'eina de generació automàtica d'analitzadors sintàctics que més us agradi: *lex i yacc* o *flex i bison* per generar codi C/C++; *JFlex i CUP* o *JFlex i BYacc/J* per generar codi Java; *PLY* per generar codi Python; *Alex i Happy* per generar codi Haskell.

3. Per a la taula de símbols, podeu utilitzar la implementació que millor s'adapti a la vostra especificació.

4. Joc de proves utilitzat per validar la solució.

5. README amb els aspectes d'ús i tots els aspectes que vulgueu destacar de la implementació i, si procedeix, de les característiques implementades de la taula de símbols.

Lliurament de la pràctica al `cv.udl.cat` dins d'activitats. Lliurar un arxiu comprimit que agrupi, tots els fitxers font, els jocs de proves utilitzats i fitxer README.

Avaluació

- La pràctica la podeu realitzar de manera individual o en grups de 2 o 3 persones.
- El pes de la pràctica és d'un 30% sobre la nota final de l'assignatura.
- A més, si presenteu els 3 exercicis obligatoris, podeu fer l'exercici opcional que s'avaluarà amb un màxim de 0,5 punts extra.
- La data límit per lliurar la pràctica és el **divendres 14 de juny** (data de finalització dels parcials), per a l'avaluació contínua. Pels que no la tingueu acabada el 14 de juny, la podeu lliurar fins al **dijous 27 de juny** (data de la recuperació de l'assignatura).