Lezione 2

20 febbraio 2012

- Linguaggio Semplice: domini Sintattici
- Caratteristiche richieste ai linguaggi
- Una Tabella sulle richieste

Linguaggio Semplice

Semplice:Domini Sintattici

T::= Int | Bool
D::= Var Ide T | D ; D
C::= Ide := E | While E do C | C ; C
E::= Ide | LV | …
Ide::= …
LV::= …

Dominio = insieme di termini + relazioni di (ordinamento)
D. Sintattico = Termini Sintattici e costruttori (sottolineati)

Caratteristiche Attese da un L.P.

1. Chiarezza, Semplicità, Unitarietà
2. Ortogonalità
3. Adatto per le Applicazioni
4. Supporto per Astrazioni
5. Facilità modifica e Riuso
6. Facilità di Verifica
7. Ambiente di Programmazione
8. Portabilità
9. Efficienza

1. Chiarezza, Semplicità, Unitarietà

Costituito da un *chiaro* [=completamente definito], *semplice* [=definito con poche regole], *unitario* [=regole espresse utilizzando uno stesso formalismo] insieme di *concetti* [=tipi, dichiarazioni, identificatori, comandi, espressioni, ..] da usare come *primitive* [= elementi di base] per esprimere in modo effettivo e formale Algoritmi/Funzioni Calcolabili.

Semplice soddisfa il requisito

2. Ortogonalità

Possibilità di combinare le features, fornite dal linguaggio,in tutti i modi possibili con ogni combinazione pienamente significativa. Poche strutture ma tante combinazioni.

Si può esagerare. Ad esempio il linguaggio C: *while E do …* può essere combinato con ogni espressione E, ottenendo sempre un programma C lecito che però non sempre calcola ciò per cui era stato scritto.

Semplice soddisfa il requisito in modo accettabile: E può essere Una qualunque espressione ma di tupo Bool.

3. Adatto per Applicazioni/1

Sintassi appropriata per riflettere la struttura dell'algortitmo:

- Strutture dati + operazioni
- Strutture di controllo

adeguate algoritmo da realizzare

Semplice soddisfa solo per sottoclasse Algoritmi su interi. Ad esempio si scriva: 1) un programma per ordinamento 2 interi; 2) Un programma per ordinamento k interi -- 2 richiede $0(k)^2$ Statements.

3. Adatto per Applicazioni/2

- Fortran/Algol/Pascal/C…: Algoritmi su Interi e a altri tipi di uso comune (array, records)
- Cobol..: Algoritmi di gestione commerciale
- ML/Haskell/Caml..: Algoritmi di manipolazione simbolica e su Tipi Astratti
- PROLOG: Algoritmi Deduttivi
- Java,Scala,C#,F#,Ocaml…: Algoritmi Object Oriented

4. Supporto per Astrazioni

Astrazioni di:

- dati
- controllo

per specifici algoritmi (anche di una stessa classe di appli cazioni) occorrono tipi di dato e forme di controllo più Adeguate, alle richieste dell'algoritmo, di quelle standard previste dal linguaggio

Esempio: Scheduling delle lezioni degli studenti nelle classi richiede dati per descrivere "studenti" e operazioni per descrivere i "comportamenti" previsti nell'algoritmo per gli studenti e la loro partecipazione alle lezioni.

Semplice non ha astrazioni (altro che sui nomi)

5. Facilità di Modifica e Riuso

Modifica: Adattabilità ai cambiamenti del contesto di uso del programma;
Riuso. Usiamo parti del programma per nuove estensioni del programma stesso o un nuovo sviluppo di esso

Modificabilità richiede che ogni porzione del programma esibisca
In modo chiaro la sua funzionalità nella definizione dell'algoritmo

Semplice non ha stutture che mostrino funzionalità di *parti* di un programma, perchè i suoi programmi non hanno parti ma sono eempre e solo un'unica sequenza di comandi

6. Facilità di Verifica

Affidabilità: strutture per il supporto di tecniche di controllo dei crash durante l'uso (ad es. type checking);
Correttezza. Strutture per il supporto di tecniche per la verifica che il programma soddisfa proprietà che sono state provate vere sull'algoritmo realizzato

Semplice può essere equipaggiato di type checker come abbiamo visto nel modulo compilatori.

7. Ambiente di Programmazione

Tutti i linguaggi di un certo valore meritano di essere equipaggiati di un ambiente di sviluppo che includa: syntax editor, debugger, tests generator, interprete, compilatore, multithread per la gestione di più strumenti, interfaccia grafica per la visualizzazione delle finestre di interazione con ogni strumento in esecuzione, …

Semplice non ha un tale ambiente

8. Portabilità

Il comportamento di ogni programma del linguaggio, mostrato in fase di sviluppo su una piattaforma, deve essere lo stesso su ogni altra piattaforma su cui il programma (il linguaggio) è eseguibile

Definizioni formali del linguaggio: Uno stesso programma C talora calcola in modo diverso se compilato con compilatori diversi.
Macchine Virtuali: Eliminano le differenze tra le varie piattaforme.

Semplice è dotato di una definizione formale

9. Efficienza

- Costo dello Sviluppo
- Costo dell'Esecuzione
- Costo del Mantenimento

Semplice: Parzialmente non applicabile perchè non ha implementazione (quindi niente esecuzione e manenimento). Circa lo sviluppo solo algoritmi banali, altrimenti lo sviluppo di venta folle: Non abbiamo array di interi ma potremmo codificare n-uple di interi con un intero, decodificando per selezionare elementi e ricodificando una volta modificati.

La Tabella di Semplice

1. Chiar., Sempl., Unitar.	soddisfa pienamente
2. Ortogonalità	ottimo compr. con correttezza
3. Applicazioni	sottoinsieme algor. su interi
4. Astrazioni	nessuna per dati e per controllo
5. Modifica e Riuso	inadatto
6. Verifica	ottimo per tipi, meno per prop.
7. Ambiente di Programmaz.	non disponibile
8. Portabilità	definizione formale
9. Efficienza	non disponibile