Introduzione al Corso di Verifica Automatica dei Sistemi: Teoria e Applicazioni

Angelo Montanari

Dipartimento di Matematica e Informatica
Università di Udine

Obiettivo

Sviluppare tecniche, metodologie e strumenti per la **specifica** e la **verifica formali** di **programmi/sistemi reattivi** (e dell'ambiente con cui essi interagiscono / che essi controllano) facendo uso della **logica temporale**

Parole chiave:

- tecniche, metodologie e strumenti formali di specifica e verifica (verifica della consistenza della specifica; verifica della validità di proprietà attese del sistema/programma)
- programmi/sistemi reattivi vs. programmi trasformazionali
- logica temporale (linguaggi logici di specifica e strumenti per la verifica della soddisfacibilità e il model checking)

Programmi/sistemi reattivi

Un **programma/sistema reattivo** è un programma/sistema il cui scopo è mantenere nel tempo (**durata indefinita**) una data modalità di interazione con l'ambiente in cui opera (anziché terminare la propria esecuzione restituendo un determinato valore finale)

Esempi di programmi/sistemi reattivi.

Sistemi operativi, programmi concorrenti e in tempo reale, programmi per il controllo di processi, programmi embedded (programmi la cui componente software è una parte integrata del sistema completo)

La concorrenza

La **concorrenza** è una componente fondamentale dei programmi/sistemi reattivi

Due forme di concorrenza (in verità possono essere ricondotte ad un'unica forma):

- un programma reattivo opera in maniera concorrente col suo ambiente
- la maggior parte dei sistemi reattivi è costituita da un insieme di processi che vengono eseguiti concorrentemente

Studieremo **tecniche formali** per la specifica e l'analisi dell'interazione fra componenti che operano concorrentemente

Modelli e linguaggi

Presenteremo un *modello computazionale*

- sistemi di transizioni equi (e un *linguaggio di programmazione* per programmi/sistemi reattivi) prima
- **automi** su oggetti infiniti (parole, alberi, grafi) poi

Useremo la **logica temporale** quale linguaggio di specifica formale

Svilupperemo **algoritmi** per la verifica della soddisfacibilità e il model checking

Comunicazione tra processi - 1

Per quanto riguarda il linguaggio di programmazione, analizzeremo i **meccanismi** per la **comunicazione** e la **sincronizzazione** tra processi concorrenti

Due modalità fondamentali di comunicazione:

- tramite **variabili condivise**
- via **scambio messaggi**

Presenteremo un **approccio uniforme** alla comunicazione tra programmi reattivi, indipendente dal particolare meccanismo/modalità di comunicazione adottato

Comunicazione tra processi - 2

Vedremo come alcuni **paradigmi fondamentali** della programmazione concorrente

- **mutua esclusione**
- **schema produttore/consumatore**

possano essere "programmati" attraverso entrambi i meccanismi/modalità di comunicazione

Studieremo anche il legame che intercorre tra **true concurrency** e **interleaving**

True concurrency vs. interleaving

True concurrency: esecuzione **realmente** concorrente dei processi

Interleaving: rappresentazione della concorrenza attraverso l'**alternanza** di azioni atomiche scelte, una per volta, dai diversi processi concorrenti (analogie con la gestione di transazioni concorrenti nelle basi di dati)

Vedremo quali **restrizioni sintattiche** sui programmi considerati e quali **requisiti di equità** (fairness) vadano imposti per consentire di utilizzare l'interleaving quale **modello fedele** della true concurrency

Proprietà dei programmi/sistemi reattivi

Prenderemo in esame le classi di proprietà più rilevanti dei programmi/sistemi reattivi

- **sicurezza** (safety)
- **vitalità** (liveness)
- **reattività** (reactivity)

Sicurezza: situazioni non desiderate non devono mai presentarsi (condizioni universali)

Vitalità: situazioni desiderate devono prima o poi realizzarsi (condizioni esistenziali)

Model checking

Model checking per le logiche temporali di base (LTL, CTL, CTL^*, μ-calculus)

Complessità computazionale del model checking (il problema dell'esplosione del numero degli stati).

Possibili soluzioni: model checking simbolico, OBDD (Ordered Binary Decision Diagram) e varianti, partial order reduction

Model checking mediante automi

Alcuni **model checker** (SPIN, SMV, NuSMV), con esempi di utilizzo

Testi di riferimento - 1

Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems: Specification, Springer, 1992

Z. Manna, A. Pnueli, Temporal Verification of Reactive Systems: Safety, Springer, 1995

A. Montanari, Sistemi Reattivi: Automi, Logiche e Algoritmi (note al corso).

W. Thomas, Automata on Infinite Objects, in Handbook of Theoretical Computer Science, Vol B (Capitolo 4), J. van Leeuwen (ed.), Elsevier, 1990.

W. Thomas, Languages, Automata, and Logic, in Handbook of Formal Languages, Vol. III, G. Rozenberg, A. Solomaa (eds.), Springer, 1997, pp. 389-455.

Testi di riferimento - 2

D. Perrin, J. E. Pin, Infinite Words: Automata, Semigroups, Logic and Games, Pure and Applied Mathematics Series, Elsevier, 2004.

A. Montanari, G. Puppis, Verification of Infinite State Systems, Note al Corso della 18th European Summer School in Logic, Language and Information (ESSLLI), Malaga, Spain, 2006.

E. A. Emerson, Temporal and Modal Logic, Handbook of Theoretical Computer Science, Vol B (Capitolo 16), J. van Leeuwen (ed.), Elsevier, 1990.

E.M. Clarke, O. Grumberg, D. Peled, Model Checking, The Mit Press, 2000.

M. Huth, M. Ryan, Logic In Computer Science: Modelling And Reasoning About Systems, Cambridge University Press, 1999.

Articoli su specifici temi del corso.