Sonic TBL: Un Percorso Sonico da Creatività a Didattica dell'Informatica

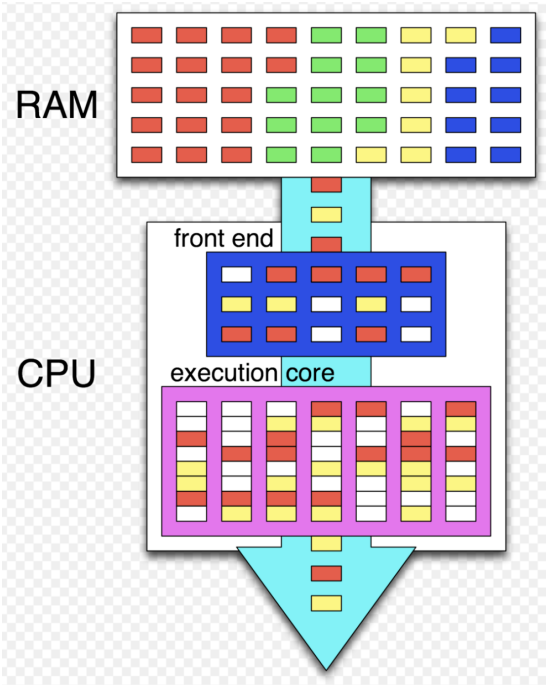
G. Delzanno, G. Guerrini, D. Traversaro

DIBRIS, Università degli Studi di Genova

ITADINFO 2023

Motivazioni

- Nella **programmazione concorrente** due o più flussi di esecuzione (thread) dello **stesso programma**, vengono eseguiti con parallelismo reale (multicore) o virtuale (scheduler) con operazioni di lettura e scrittura su memoria condivisa.
- La concorrenza introduce possibili conflitti (data race) e non determinismo (l'ordine di esecuzione delle istruzione non è noto a priori)



Motivazioni

- A causa della sempre crescente rilevanza del multi-threading (*), scuole ed università hanno iniziato a introdurre la programmazione concorrente anche nei corsi di introduzione alla programmazione
- Gli studenti hanno comunque difficoltà a comprendere la concorrenza in profondità (al di là dell'uso di pattern per l'uso di librerie), compito ancor più impegnativo se intrapreso nei primi passi verso la programmazione.

(*) *da wikipedia:*

Most forecasters, including Gordon Moore,[120] expect Moore's law will end by around 2025.[121][118][122]

For years, processor makers delivered increases in clock rates and instruction-level parallelism, so that single-threaded code executed faster on newer processors with no modification.[140] Now, to manage CPU power dissipation, processor makers favor multi-core chip designs, and software has to be written in a multi-threaded manner to take full advantage of the hardware. Many multi-threaded

The Free Lunch is Over

`http://www.gotw.ca/publications/concurrency-ddj.htm`

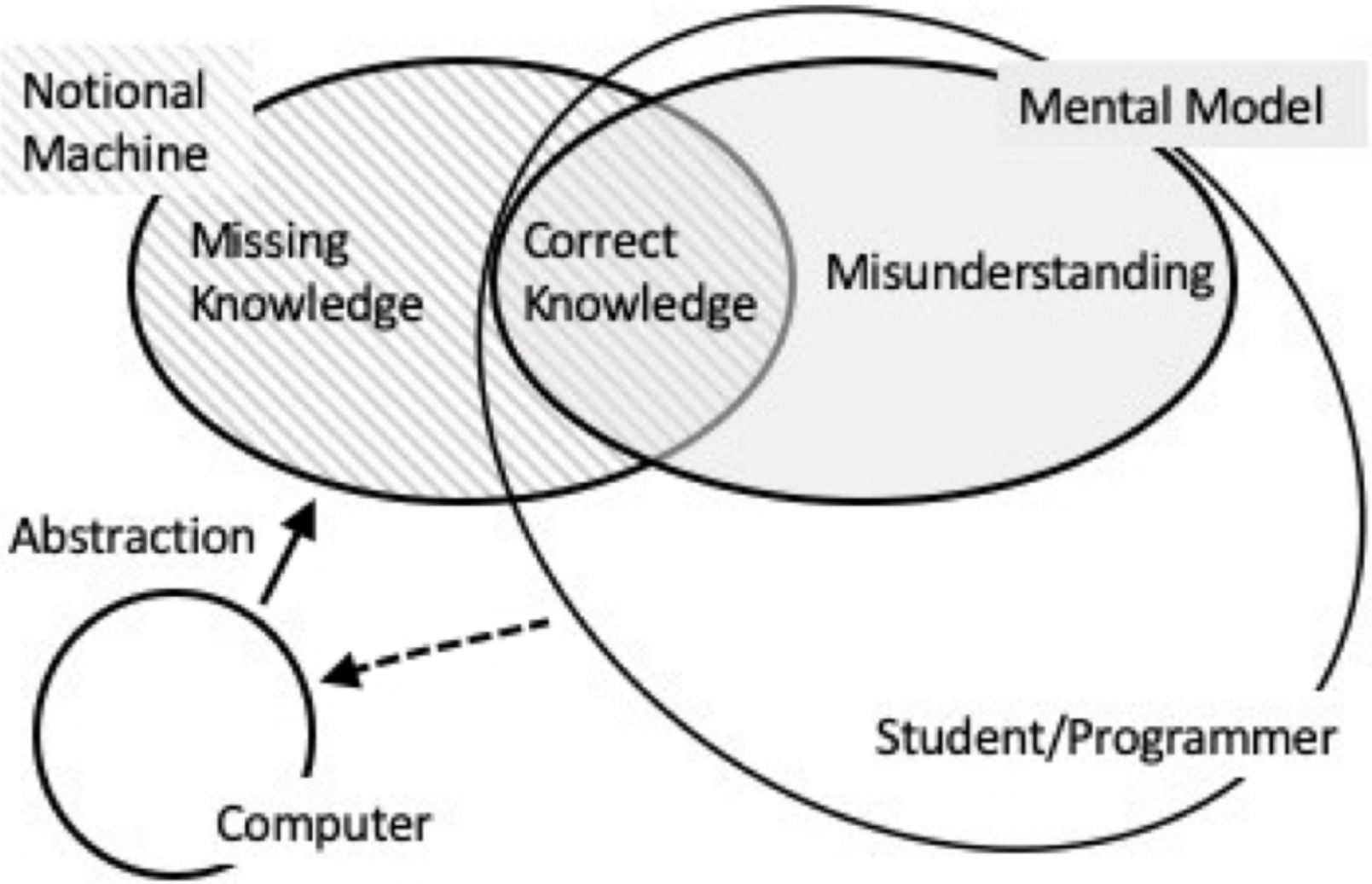
Concurrency Education

- La concurrency education si concentra sull'analisi del modello mentale che gli studenti sviluppano quando affrontano la programmazione concorrente
- Uno degli obiettivi è in particolare quello di identificare fraintendimenti ed errori comuni

Mental Model vs Notional Machine

- I modelli mentali possono essere visti come una rappresentazione cognitiva della realtà (come uno studente interpreta un concetto)
- Gli studenti tipicamente costruiscono un loro modello mentale di una data macchina nozionale (astrazione di una macchina reale, es elaboratore, runtime system, ecc)
- La macchina nozionale viene usata per descrivere concetti ed aspetti operazionali della macchina reale
- Capire la macchina nozionale è un obiettivo di apprendimento fondamentale in particolare per beginner

Misconception



Esempi di Concurrency Misconception

- I programmi concorrenti devono funzionare nella maggior parte dei casi, non serve quindi considerare tutti i possibili interleaving delle loro istruzioni
- I lock vengono interpretati come una possibile protezione di parti di codice invece che di risorse condivise (il modello mentale si focalizza sul codice invece che sui dati)
- Confusione nell'uso di variabili globali, automatiche e thread local
- Le primitive di sincronizzazione (lock, semafori, metodi sincronizzati, ecc) assicurano sempre la correttezza nell'accesso alle risorse indipendentemente dalla granularità con cui vengono adottate
- Si assume che la macchina nozionale soddisfi consistenza forte o sequenziale, dimenticandosi quindi che le architetture moderne sono spesso basate su modelli rilassati

Modelli astratti della concorrenza

Esistono molti linguaggi astratti per rappresentare e ragionare su computazioni concorrenti, ad esempio:

- Petri Nets
- Communicating Automata
- Process calculi (CCS, CSP)

Modelli astratti della concorrenza

Esistono molti linguaggi astratti per rappresentare e ragionare su computazioni concorrenti, ad esempio:

- Petri Nets
- Communicating Automata
- Process calculi (CCS, CSP)

Maneggiare con cautela

Adatti a studenti con buone capacità di astrazione, dote che però che mediamente si acquisisce nel tempo.

Approcci più concreti

- Modelli computazionali in cui la concorrenza è un concetto naturale: videogiochi, simulazioni, musica, ecc.
- Servono comunque linguaggi e strumenti per allenare il nostro modello mentale, esplorare misconception che poi si incontrano nella programmazione e trasferire conoscenza verso linguaggi di programmazione tradizionali
- I **linguaggi di programmazione domain-specific** possono essere di grande aiuto!

Nostra proposta

Active learning per studenti della triennale di informatica con **Sonic Pi**, un linguaggio di programmazione per live music coding

Perchè Sonic Pi?

- Sonic Pi è un linguaggio "da grandi" basato su Ruby (es supporta hot swapping)
- Essendo nato come linguaggio per programmazione creativa, si combina bene con metodi innovativi (flipped-class, tean work) Si possono usare dichiarazioni annidate per definire pipeline di effetti sonori
- Supporta il multithreading (buffer multipli e thread concorrenti) con astrazioni semplificati ed con una semantica intuitiva per la temporizzazione degli eventi
- Permette di "ascoltare" errori e far "risuonare" misconception

I nostri esperimenti in 3 diversi anni accademici

1. Introduzione alla Programmazione (1 anno LT):
   Concetti di base del multithreading, focus su creazione di brani musicali e team work
2. Programmazione concorrente (III anno LT):
   Concetti di base del multithreading e uso di API Python, linguaggio usato in diverse attività di didattica innovativa
3. Architetture dei Calcolari (I anno LT):
   Misconception su concetti di base (3 livelli di visibilità delle dichiarazioni, ecc)
   Programmazione concorrente (III anno LT): Focus su concurrency misconception su concetti di base e avanzati (es. operazioni thread safe su strutture dati)

Metodologia

1. Pre-class material (una settimana prima delle attività in classe)
2. RAT Quiz: Domande a risposta multipla per prendere confidenza con Sonic Pi (sintassi, editor, esempi, ecc), sia individuale che a gruppi (per valutare impatto del lavoro in team)
3. Team application tasks: diversi tipi di esercizi (comprensione ma anche scrivere script per Sonic Pi).
   Ogni esercizio focalizzato su specifiche misconception della programmazione concorrente usando appunto i costrutti più semplici forniti da Sonic Pi e la corrispondente semantica uditiva, attività in gruppo

Esempio di RAT Quiz

Considerate il seguente programma Sonic Pi:

```
live_loop :foo do
    sample :ambi_choir
    sleep 0.5
end
in_thread do
    sample :ambi_drone
end
```

Cosa succede quando viene eseguito?

- `live_loop` viene riprodotto ripetutamente, mentre `in_thread` una sola volta
- `in_thread` e `live_loop` non possono essere eseguiti simultaneamente
- `in_thread` e `live_loop` vengono eseguiti entrambi ripetutamente
- Solo `live_loop` viene eseguito.

Esempio di Team App Task 1

A

```
in_thread do
  use_synth :piano
  x = 40
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

in_thread do
 use_synth :kalimba
  x = 40
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

B

```
x = 40
in_thread do
  use_synth :piano
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

in_thread do
  use_synth :kalimba
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

C

```
x = 40
in_thread do
  use_synth :piano
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

x = 60
in_thread do
  use_synth :kalimba
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

Example of Team App Task 2

Voting cards: Quale risposta è corretta?

- In B e C threads differenti operano su una risorsa comune e il suono prodotto dipende dall'ordine di esecuzione delle loro istruzioni
- In A e C threads differenti operano su una risorsa comune e il suono prodotto dipende dall'ordine di esecuzione delle loro istruzioni
- In nessuno dei programmi ci sono risorse condivise tra thread
- In A e C non ci sono risorse condivise tra thread e il suono prodotto dipende dall'ordine di esecuzione delle loro istruzioni

Hear and Play Misconceptions

```
use_synth :piano

note=[52,55,59,40]
i=0

define :foo do |x|
  in_thread do
     play note[x]
  end
end

3.times do
  foo i
  i+=1
end
```

Voting cards: Which answer is true?

(1) The program has no race conditions.
(2) The program creates only one thread.
(3) The program plays the notes in the "`note`" list in sequence.
(4) The program plays the E minor chord.

Gallery walk:
Describe in detail the behavior of the script with particular pay attention to the changes to the value of the variable "`i`" and to the possible sequences of notes play.

Conclusioni

- Experimenti con 184 studenti nel 2022/23
  - Abbiamo provato ad allenare il (nostro ed il loro) modello mentale con strumenti che sono risultati interessanti anche per informatici "nudi e crudi"
  - Abbiamo fatti primi passi per knowledge transfer da Sonic Pi a C/Java
- Feedback molto positivo (4 su scala Likert 1-5) dei partecipanti che hanno compilato i questionari (39 su 184)
- Abbiamo individuato nuovi tipi di concurrency misconception che non dipendono strettamente dal linguaggio (es 3 livelli di visibilità)
- Risultati promettenti e buone indicazioni sulle pratiche utilizzate:
  - attività individuale e di gruppo su stessi task
  - gruppo di controllo "sonico" monitorato negli esami