

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
Scuola di Ingegneria e Architettura
Corso di Laurea in Ingegneria Informatica

Attualizzazione del supporto al multi-threading in tuProlog

Tesi di Eleonora Cau

Relatore Prof. Enrico Denti

Anno Accademico 2011/2012

CONTESTO

- tuProlog è un interprete Prolog scritto in Java, attualmente giunto alla versione 2.6.
- Esiste un prototipo multi-threaded dell'interprete tuProlog sviluppato per la versione 2.4 e basato sul modello di parallelismo esplicito.

OBIETTIVI

- Dotare l'interprete di un **supporto per il multi-threading**
- **Studio di fattibilità:**
 - Integrazione tra il precedente prototipo e l'attuale motore
 - Sviluppo del nuovo prototipo a partire da un approccio diverso

ANALISI DEL PROBLEMA

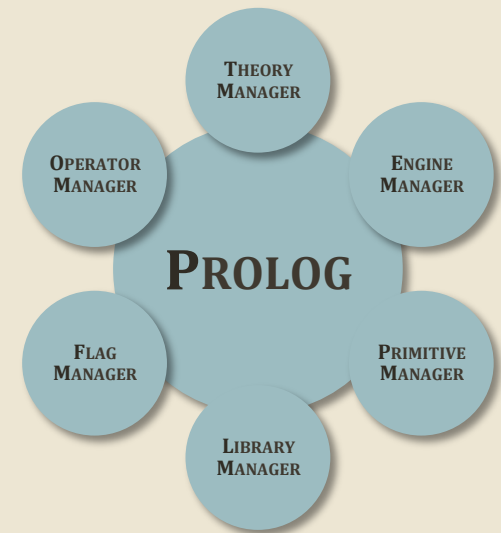
- Violazione di:
 - concetto di separazione delle responsabilità tra i manager
 - ruolo di mediatore del motore inferenziale Prolog
 - principi di progettazione
- Presenza di relazioni errate, a livello logico, tra le classi
- Generazione di thread non necessari: inefficienza
- Gestione della modalità sequenziale malfatta
- Architettura poco lineare e ridondante



Necessaria RIPROGETTAZIONE

THREAD-SAFETY: L' INFRASTRUTTURA

- Sincronizzare l'**ambiente di esecuzione** per garantire una **corretta competizione** tra i processi:
 - evitare condizioni di deadlock
 - garantire la consistenza dei dati
- Occorre intervenire su:
 - i **manager** che includono le informazioni condivise e interagiscono con il resto del sistema
 - i **mediatori** che sono il mezzo di comunicazione fra i manager e le librerie

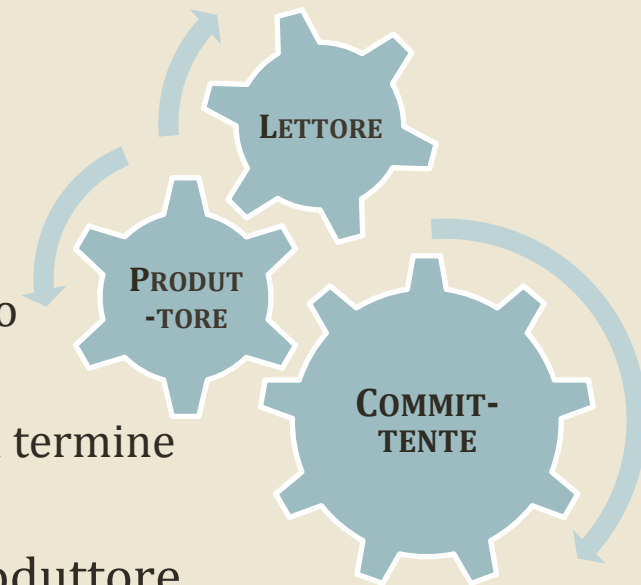


THREAD-SAFETY: IL MODELLO

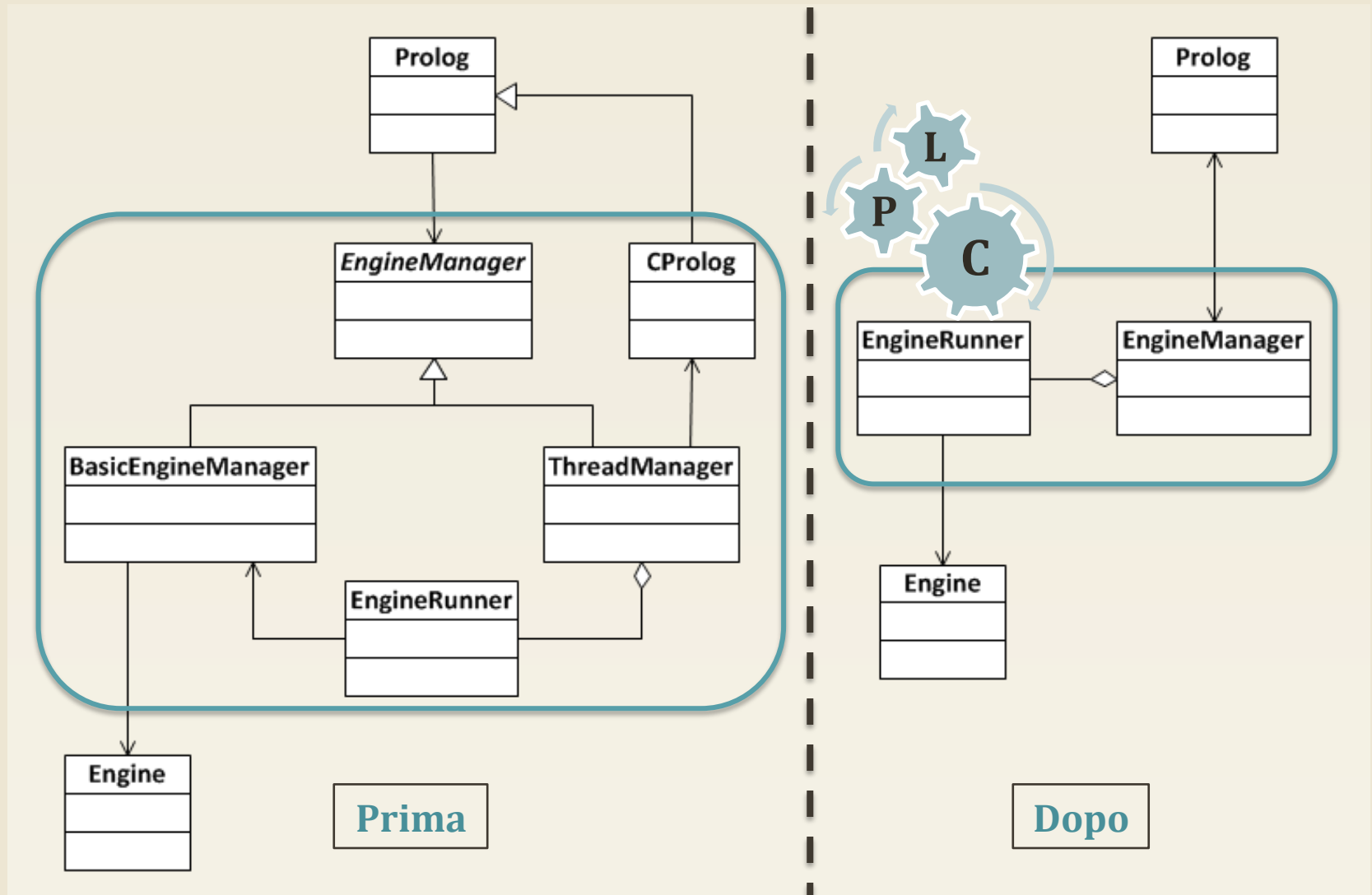
➤ Garantire una **corretta cooperazione** attraverso l'imposizione di vincoli per le interazioni tra i processi.

➤ Definizione di ruoli:

- il **lettore** preleva le soluzioni
 - Deve attendere che il produttore abbia finito di computare.
 - Ha sempre la precedenza sui committenti al termine di una computazione del produttore.
- il **committente** riavvia l'esecuzione del produttore
 - In qualunque momento, può depositare la richiesta per una computazione successiva e proseguire con la sua esecuzione.
- il **produttore** sta risolvendo o ha risolto una query
 - Non può compiere alcuna operazione mentre sta computando.



ARCHITETTURE A CONFRONTO



THREADLIBRARY

Supporto alla programmazione concorrente.

➤ Definisce numerose primitive per:

- la **creazione** e la **distruzione** di thread (`thread_create/2...`)
- il **prelievo** e la **ricerca** di nuove soluzioni (`thread_join/2`, `thread_next_sol/2...`)
- la **comunicazione** (`thread_send_msg/2`, `thread_get_msg/2...`)
- la **mutua esclusione** (`mutex_create/1`, `mutex_lock/1...`)
- predicati **backtrackabili** (`thread_execute/2`)

COLLAUDO

- Funzionamento dei predicati della ThreadLibrary
- Correttezza dell'infrastruttura e del modello di sincronizzazione
- Retrocompatibilità
- Efficienza del modello multi-threaded

Fase di collaudo prestazionale: confronto tra la versione sequenziale di tuProlog 2.6 e il nuovo prototipo

Esempio	$T_{seq.} (s)$	$T_{conc.} (s)$	$\Delta T (s)$	Guadagno %
Calcolo del fattoriale	0,034	0,039	-0,005	-14%
Calcolo della media dei voti	0,70	0,65	+0,050	+8%
Manipolazione di liste	0,134	0,112	+0,022	+16%

CONCLUSIONI

Nuovo supporto per il multi-threading in tuProlog:

- Differenze strutturali e concettuali rispetto al precedente prototipo
- Maggiori funzionalità a disposizione dell'utente
- Incremento delle prestazioni al confronto con il modello sequenziale

Vi ringrazio per l'attenzione.