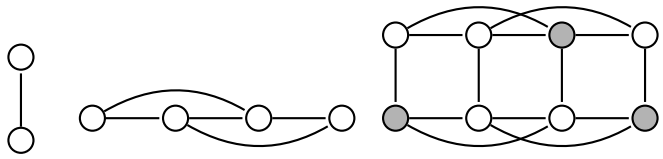
RIASSUNTO

1. Stable Sets of SuperGrid Graphs via Finite State Automata

Gli *insiemi indipendenti* (o anche *insiemi stabili*) di un grafo G sono sotttoinsiemi di vertici di G che non contengono vertici adiacenti. Queste strutture hanno un grande interesse in Informatica se non altro per il motivo che determinare se un grafo possiede un insieme indipendente di data cardinalità è un problema NP-completo.

Questa Tesi si occupa della enumerazione degli *insiemi indipendenti* (o anche *insiemi stabili*) di varie classi di grafi. Poche definizioni per illustrarne una che abbiamo chiamato classe dei *grafi supergriglia* o, semplicemente, delle *supergriglie*. Una supergriglia è un *prodotto cartesiano* di *potenze di cammini*. La potenza h-esima del cammino P_n è un grafo, $P_n^{(h)}$, con n vertici ognuno dei quali è collegato agli h vertici successivi. Quindi una supergriglia è un grafo individuato da quattro parametri: le lunghezze dei due cammini e le loro potenze.

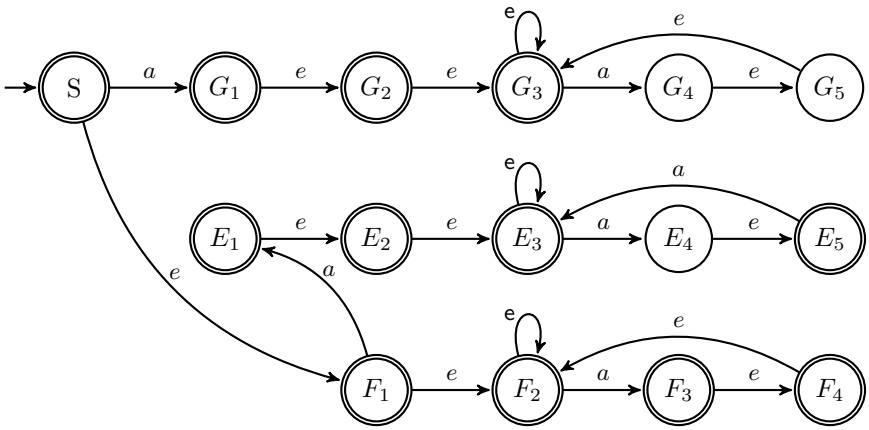
La figura qui sotto mostra, da sinistra a destra, $P_2^{(1)}$, $P_4^{(2)}$ e un 3-insieme indipendente della SuperGriglia $P_2^{(1)} \times P_4^{(2)}$.



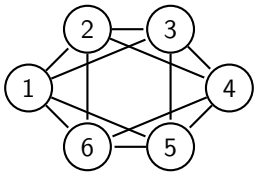
L'originalità della Tesi sta nell'approccio al problema: *per ognuna delle classi considerate, siamo in grado di costruire una automa a stati finiti che riconosce le stringhe che corrispondono agli insiemi indipendenti dei grafi della classe in esame.*

In tal modo siamo in grado di enumerare gli insiemi indipendenti di grafi (circuiti, $C_n^{(h)}$, e loro prodotti cartesiani) per i quali non si possono applicare le tecniche utilizzate in letteratura [Wilf]. Ecco un esempio: questo automa

Date: 5 novembre 2015.
Key words and phrases. IntroGAL.tex.



riconosce (tutte e sole) le stringhe che descrivono gli insiemi indipendenti di $C_6^{(2)}$:



I nostri risultati sono ottenuti utilizzando il linguaggio *Java* e l'interazione con il software *Wolfram Mathematica* tramite protocollo *MathLink*. L'ambiente di sviluppo è *Wolfram Workbench 2.0*. Le tecnologie di supporto impiegate nella costruzione del programma sono *Spring Framework 3.1*, *JUnit* per i test unitari e *Velocity* per la creazione di documenti pdf da specifico template.

Altre librerie impiegate sono: *combinatoricslib-2.0*, *log4j-1.2.17* per salvare su file di log i passi di esecuzione di ogni routine del programma, *oeis-java-sdk* libreria che si interfaccia con il sito *oeis.org* e che, data in input una successione, ritorna come output i risultati del database di Sloane.

Il programma (costituito da circa 70 classi java per un totale di 13.200 righe di codice) è stato costruito utilizzando il pattern *MVC (Model View Controller)* dove per la parte controller è stato creato un prompt di comando che riceve e interpreta comandi digitati dall'utente ed esegue le routine richieste.

Contestualmente al pattern MVC è stato realizzato il pattern *Singleton* che ha lo scopo di garantire che la classe per la gestione delle connessioni al kernel di Mathematica venga istanziata una sola volta per tutta la durata di esecuzione dell'applicazione. Tale classe fornisce inoltre un punto di accesso globale verso tutte le classi java, similmente a quanto avviene con le connessioni al database per applicazioni *Web Oriented*.

Per stampare i disegni dei grafi su pdf è stato utilizzato il plugin *graphviz* di *LaTex* in combinazione con le potenzialità di *dot2tex*, altro strumento utile per semplificare la sintassi di costruzione di un grafo e che genera la figura corrispondente e la incolla al documento.

Per poter eseguire il software è necessaria l'installazione di *Mathematica*, del compilatore *LaTex* e del programma *Graphviz*.