generate_NOproofs4NPproperties

Breve descrizione del progetto
Il programma vuole aiutare un soggetto umano nel tentativo di ottenere una dimostrazione procedendo per casi, tenendo memorizzati i casi e sotto-casi aperti. Ad esempio, il progetto può essere utilizzato per produrre una dimostrazione che un controesempio è di fatto un controesempio per un dato problema.
L'utente può costruire l'albero di esplorazione, creando un numero di figli a piacere, dove può imporre dei vincoli per cercare di ottenere la sua dimostrazione. Ovviamente, il sistema avvisa l'utente se i figli violano i vincoli del modello.

Alla base del progetto, vi è il linguaggio Minizinc, pertanto è necessario che l'utente conosca il linguaggio di modellazione appena indicato e i concetti base di constraint programming. Infatti, qualora un modello non fosse specificato correttamente, il programma non permetterebbe di inizializzare l'albero di esplorazione delle soluzioni.
Maggiori informazioni su come modellare i problemi possono essere recuperate sul sito ufficiale: (https://www.minizinc.org/).

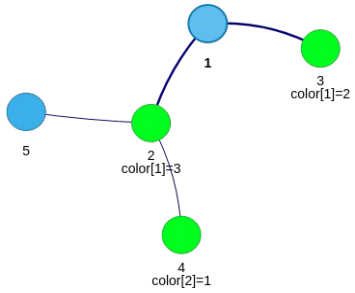
Il sistema offre un sistema di visualizzazione nel browser dell'albero sviluppato e genera su richiesta dell'utente un Report in PDF che mostra gli step della dimostrazione, ossia i vincoli inseriti per il branch che si sta esplorando.

Requisiti
Di seguito vengono elencati i requisiti di sistema per il corretto funzionamento del programma.
Il programma è eseguibile da terminale, lanciando il comando p*ython3 \ python main.py.* Affinché tutto funzioni correttamente sono necessari i seguenti requisiti:

1. Ambiente Linux. In Windows si sono presentate delle problematiche con l'interfaccia Python di Minizinc. Non è stato testato in ambiente Mac.
2. Linguaggio di programmazione Python
3. Minizinc Python: interfaccia di Python per gestire e modellare con Minizinc - si tratta di un progetto open source e ancora in via di sviluppo. (https://minizinc-python.readthedocs.io/en/latest/getting_started.html)
4. Devono essere inoltre installate le seguenti librerie Python sul sistema (sempre disponibili tramite comando pip):
    1. Pyvis
    2. fpdf
    3. imgkit
    4. tkinter

Modello di esempio
Modello: graph_col.mzn
Il file descrive il problema della colorazione propria di un grafo.
In base all'istanza specificata nel file, il modello genererà le possibili soluzioni.
L'utente può creare il proprio albero di esplorazione delle soluzioni per cercare di verificare un controesempio; più nello specifico, ogni figlio rappresenta un possibile sottocaso dell'albero di esplorazione.
Una volta creati i branch (ad esempio 2 figli a partire dalla radice), l'utente può specificare dei vincoli, come:
- constraint color[1] = 3 per il figlio 1
- constraint color[1] = 2 per il figlio 2

Questo equivale a voler verificare due controesempi della colorazione propria aventi come colore per il vertice 1 il colore 3 e il colore 2. L'utente può quindi procedere e generare tutti i sottocasi che desidera, a patto che non vengano violati i vincoli del modello.



I nodi privi di assegnamento sono colorati di azzurro, mentre i nodi con assegnamento verificato dal modello (cioè che non violano il modello stesso) sono in verde. Si consiglia di lasciare sempre vuoto l'assegnamento della radice, ossia del nodo 1, in modo da generare più sottocasi. Infatti, ogni figlio eredita il modello e in vincoli dal padre; di conseguenza, assegnare un vincolo alla radice impone di verificare un solo controesempio alla volta, mentre se si creano dei figli è possibile esplorare l'albero delle soluzioni sia in ampiezza che in profondità.
Qualora si voglia chiudere un nodo foglia, ad esempio perché non ci sono soluzioni, è possibile commentare il nodo stesso con una assegnamento scelto dall'utente. Questo non verrà considerato nel modello e il programma provvederà a colorare di nero il nodo foglia chiuso. **Quando un nodo viene chiuso non deve essere più considerato per la dimostrazione.**