Hlavní funkce programu

- SAT solver formulí v CNF – jak paralelní, tak sekvenční
- Řešení některých dalších NP-úplných problémů převodem na SAT a použitím solveru
  - Problém nezávislé množiny v grafu
  - Problém obarvení grafu třemi bravami
  - Problém hamiltonovské cesty

Hlavní řešené problémy

- Implementace DPLL algoritmu
  - a převod z rekurzivního na iterační kvůli přetečení zásobníku
- Paralelizace DPLL algoritmu
  - a jak rovnoměrně rozdělit práci mezi vlákna + synchronizace
- Převody dalších problémů na SAT a interpretace výsledků

Nástin architektury

- Třída *CNF* reprezentující formuli / (parciální) model
  - Obsahuje seznam klauzulí (typ *Clause*)
    - Typ *Clause* obsahuje seznam proměnných
  - Instanční metody *void ReadFormula(TextReader reader)*, *string ToString(), string GetInputFormat()*
- Třída *DPLL* reprezentující instanci algoritmu
  - Instanční metody *DPLLResultHolder Satisfiable(CNF cnf)*, *DPLLResultHolder SatisfiableParallel(CNF cnf)*
- Třída *DPLLResultHolder* reprezentující výsledek SAT solveru
  - Instanční metoda *string ToString()*
- Třídy pro každý další problém
  - Instanční metody *void ReadInput()*, *CNF ConvertToCNF()*, *string InterpretDPLLResult(DPLLResultHolder result)*

Nástin architektury

- Paralelismus
  - Varianta modelu producent-konzument
    - Vytvoření pracovních vláken (tolik, kolik je k dispozici logických procesorů)
    - Každé z nich hraje roli jak konzumenta, tak producenta
    - Hlavní vlákno spí, zatímco čeká na výsledek
    - Každé vlákno v místě větvení vytvoří nový CNF model, v němž vybranou proměnnou nastaví nějak a v původním modelu opačně.
      - Nový model je zařazen do sdílené fronty. Na původním vlákno pokračuje.
      - Vlákno, které nemá co na práci si z fronty práci vezme.
    - Najde-li vlákno řešení, ukončí ostatní vlákna a probudí hlavní
    - Je-li sdílená fronta prázdná a žádné vlákno nepracuje, formule není splnitelná