VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Komponenta výukového serveru TI - P-úplné problémy

Component of Teaching Server for Theoretical Computer Science - Pcomplete problems

Tomáš Kirnig

Bakalářská práce

Vedoucí práce: Ing. Martin Kot, Ph.D.

Ostrava, 2025

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/25/116933
vsboee25037FA8

Zadání bakalářské práce

Student: **Tomáš Kirnig**

Studijní program: B0613A140014 Informatika

Téma: Komponenta výukového serveru TI - P-úplné problémy

Component of Teaching Server for Theoretical Computer Science - P-complete problems

Jazyk vypracování: čeština

Zásady pro vypracování:

V rámci diplomových a bakalářských prací vzniká výukový server pro předměty teoretické informatiky. Jedná se o sadu dynamických webových stránek umožňujících studentům pochopení různých typů úloh a problémů. Na rozdíl od běžných výukových textů s pevně daným počtem ukázkových příkladů umí tyto stránky generovat libovolně mnoho ukázek na základě vstupů od uživatele. Cílem této konkrétní bakalářské práce je vytvořit komponentu pro pomoc s výukou tzv. P-úplných problémů.

Vytvořte dynamické webové stránky umožňující uživateli následující:
1. Simulovat výpočet řešení problému Monotone Circuit Value Problem (MCVP) a alespoň 2 dalších P-úplných problémů.
2. Vstupy těchto algoritmů bude moci uživatel zadávat třemi způsoby:
a) Vhodným, uživatelsky přívětivým, způsobem ručně.
b) Nechat si vstup vygenerovat zcela náhodně podle nastavených parametrů.
c) Vybrat z předpřipravené sady vhodně zvolených vstupů.
3. Bude možné si zobrazit převod instance problému MCVP na ty dva zvolené P-úplné problémy. Přitom:
a) Instanci MCVP pro převod bude možné zadat kterýmkoliv z výše uvedených způsobů.
b) Převod si bude moci uživatel krokovat se zobrazením slovního vysvětlení jednotlivých kroků.
c) Na převodem vytvořenou instanci bude opět možné použít výše požadovanou simulaci výpočtu řešení.

Seznam doporučené odborné literatury:

[1] Miyano, S., Shiraishi, S., Shoudai, T.: "A List of P-Complete Problems", Kyushu University, RIFIS-TR-CS-17, December 1990, dostupné z URL: https://catalog.lib.kyushu-u.ac.jp/opac_download_md/3123/rifis-tr-17.pdf
[2] Sawa, Z.: "Teoretická informatika", podklady pro přednášky, VŠB - Technická univerzita Ostrava, dostupné z URL: https://www.cs.vsb.cz/sawa/ti/slides/ti-slides-03.pdf
[3] Papadimitriou, C.: Computational Complexity, Addison Wesley, 1993
[4] Arora, S., Barak, B.: Computational Complexity: A Modern Approach, Cambridge University Press, 2009

17. listopadu 2172/15
708 00 Ostrava-Poruba
Česká republika

spojovatelka: +420 597 321 111
epodatelna: epodatelna@vsb.cz
ID datové schránky: d3kj88v

IČ: 61989100
DIČ: CZ61989100

email: studijni.fei@vsb.cz
www.fei.vsb.cz

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Datum zadání: 01.09.2024

Datum odevzdání: 30.04.2026

Garant studijního programu: Ing. Tomáš Fabián, Ph.D.

V IS EDISON zadáno: 18.08.2025 09:42:23

Abstrakt

Tato bakalářská práce se zabývá vývojem výukové webové aplikace pro demonstraci P-úplných problémů. Cílem je usnadnit studentům pochopení a procvičování těchto problémů. Aplikace se zaměřuje na tři P-úplné problémy: *Monotone Circuit Value Problem* (MCVP), *problém neprázdnosti bezkontextové gramatiky* a *kombinatorické hry na grafech*. Práce zahrnuje implementaci modulů pro interaktivní zadávání vstupů (ručně, náhodnou generací nebo výběrem z připravené sady) a simulaci jejich řešení. Uživatel může také sledovat krokový převod instance MCVP na další zmíněné problémy a následně jejich řešení.

Klíčová slova

Teoretická informatika, P-úplné problémy, Monotone Circuit Value Problem, webová aplikace, simulace, převod instancí

Abstract

This bachelor's thesis focuses on the development of a teaching-oriented web application for illustrating P-complete problems. The main goal is to facilitate students' understanding and practice of such tasks. The application focuses on the *Monotone Circuit Value Problem* (MCVP), *Empty Grammar*, and the *Combinatorial Game*. The project implements modules for interactive input of problem instances (manually, via random generation, or by selecting from a pre-defined set) and provides a simulation of their solutions. Users can also observe a step-by-step reduction from an MCVP instance to the other mentioned problems and subsequently explore how those are solved. The resulting application demonstrates key concepts of theoretical computer science, including the notion of P-completeness, and provides a flexible basis for educational use.

Keywords

Theoretical computer science, P-complete problems, Monotone Circuit Value Problem, web application, simulation, instance reduction

Obsah

Seznam použitých zkratek a symbolů

MCVP	–	Monotone Circuit Value Problem
P	–	Třída problémů řešitelných v polynomiálním čase
NC	–	Nick's Class – třída efektivně paralelizovatelných problémů
BFS	–	Breadth First Search – Algoritmus průchodu do šířky
CFG	–	Context-Free Grammar – Bezkontextová gramatika
DAG	–	Directed Acyclic Graph – Orientovaný acyklický graf
DFS	–	Depth First Search – Algoritmus průchodu do hloubky
DOM	–	Document Object Model – Objektový model dokumentu
HTML	–	Hypertext Markup Language – Hypertextový značkovací jazyk
ID	–	Identifier – Identifikátor
JSON	–	JavaScript Object Notation – Objektový zápis JavaScriptu
LS	–	Levá strana
PS	–	Pravá strana
SPA	–	Single Page Application – Jednostránková webová aplikace
TI	–	Teoretická informatika
UI	–	User Interface – Uživatelské rozhraní

Seznam obrázků

Kapitola 1

Úvod

Teoretická informatika poskytuje způsob, jak zkoumat, porozumět a optimalizovat možnosti a limity výpočetních systémů. Hlavním bodem tohoto zkoumání je teorie složitosti, která klasifikuje problémy na základě zdrojů potřebných pro jejich vyřešení, jako je čas a paměť. Jednou z nejvýznamnějších složitostních tříd je třída P, zahrnující problémy, řešitelné v polynomiálním čase na deterministickém Turingov stroji [9] – tedy takové, jejichž časovou složitost lze vyjádřit jako $O(n^k)$ [10] pro nějakou konstantu k, kde n je velikost vstupu – na deterministickém Turingově stroji [9]. Přestože jsou problémy v této třídě obvykle pokládány za „efektivně řešitelné“, projevují se mezi nimi významné odlišnosti, zejména pokud začneme řešit jejich paralelizovatelnost.

V tomto kontextu hraje klíčovou roli podmnožina *P-úplných* problémů (*P-complete* problems). P-úplné problémy představují výpočetně nejnáročnější úlohy v rámci třídy P. Jsou to problémy, na které lze s logaritmickou paměťovou složitostí převést jakýkoliv jiný problém z třídy P [6, 1]. Hlavní problém u této podmnožiny spočívá v předpokladu, že P-úplné problémy pravděpodobně nelze efektivně paralelizovat. To znamená, že tyto úlohy nespadají do třídy NC, která obsahuje problémy řešitelné v polylogaritmickém čase $O(\log^k n)$ pomocí paralelního výpočtu s polynomiálně mnoha procesory [7]. Z toho vyplývá, že na rozdíl od NC problémů řešení P-úplných problémů vyžaduje sekvenční zpracování. Studium P-úplnosti nám tak pomáhá vymezit hranici mezi paralelizovatelným a čistě sekvenčním výpočtem.

Tento projekt představuje interaktivní ukázku konceptu P-úplnosti prostřednictvím webové aplikace. Cílem je vytvořit nástroj, který umožní uživatelům vizualizovat a pochopit výpočet a převod mezi zvolenými P-úplnými problémy.

Jako hlavní problém byl zvolen *Monotone Circuit Value Problem* (MCVP), ve kterém se vyhodnocuje logický obvod tvořený pouze hradly typu AND a OR [7]. Pro demonstraci univerzálnosti konceptu P-úplnosti aplikace implementuje také simulace dvou dalších problémů:

- **Kombinatorické hry na grafu:** Úloha analyzující existenci vítězné strategie v deterministické hře dvou hráčů [7].

- **Problém neprázdnosti jazyka bezkontextové gramatiky:** Rozhoduje, zda daná gramatika generuje neprázdný jazyk [7].

Hlavním přínosem vytvořené aplikace je možnost vizualizace nejen samotného řešení těchto úloh, ale především *převodů* (redukcí) mezi nimi. Uživatel může sledovat krokovou transformaci instance MCVP na instanci hry nebo gramatiky, což názorně ilustruje princip polynomiálních redukcí a vzájemnou převoditelnost P-úplných problémů [7].

Výsledkem práce je webová aplikace navržená jako flexibilní nástroj pro výuku. Umožňuje uživatelům pracovat s vlastními vstupy, generovat náhodné instance pro testování a využívat předpřipravené sady úloh.

Text práce je členěn do několika částí. Po úvodu následuje kapitola věnovaná použitým technologiím a architektuře aplikace. Jádro práce tvoří tři kapitoly, z nichž každá se detailněji věnuje jednomu z implementovaných problémů: nejprve Monotone Circuit Value Problem (MCVP), následně kombinatorické hry na grafu a nakonec problém neprázdnosti jazyka bezkontextové gramatiky. Závěr práce shrnuje dosažené výsledky a navrhuje možnosti dalšího rozšíření.

Kapitola 2

Teoretický základ

Tato kapitola se věnuje teoretickému rámci, který tvoří základ této práce. Nejprve zavedeme klíčové pojmy z teorie složitosti a následně definujeme tři P-úplné problémy, kterými se aplikace zabývá.

2.1 Základní pojmy a definice

Pro formální ukotvení problematiky nejprve zavedeme klíčové pojmy z teorie složitosti, o které se tato práce opírá.

Definice 1 (Třída P) *Třída **P** (Polynomial time) obsahuje všechny rozhodovací problémy, které jsou řešitelné na deterministickém Turingově stroji v čase $O(n^k)$, kde n je velikost vstupu a k je nezáporná konstanta.*

Definice 2 (Logaritmická redukce) *Nechť A a B jsou jazyky (problémy). Řekneme, že A je **převeditelný v logaritmickém prostoru** na B (značíme $A \leq_L B$), jestliže existuje funkce f vyčíslitelná Turingovým strojem s logaritmickou paměťovou složitostí taková, že pro každé slovo w platí:*

$$w \in A \iff f(w) \in B$$

Definice 3 (P-úplnost) *Problém A se nazývá **P-úplný**, jestliže platí dvě podmínky:*

1. *$A \in P$ (problém je v třídě P).*
2. *Pro každý problém $B \in P$ platí $B \leq_L A$ (každý problém z P lze na A převést v logaritmickém prostoru).*

2.2 Monotone Circuit Value Problem

Monotone Circuit Value Problem je základní problém v teorii složitosti, který patří mezi P-úplné problémy [7]. Jeho definice je následující:

- **Vstup:** Booleovský obvod bez negací, tvořený pouze hradly AND ($\wedge$) a OR ($\vee$), společně se vstupními hodnotami (pouze hodnoty 0 nebo 1 - false nebo true).
- **Výstup:** Hodnota výstupního uzlu obvodu (výstupního hradla).

Obvod lze reprezentovat jako orientovaný acyklický graf (DAG), kde uzly představují buď vstupní proměnné (listy) nebo logická hradla (vnitřní uzly). Hrany reprezentují tok logických hodnot – přenášejí výsledky vyhodnocení z jednoho uzlu jako vstupy do uzlů následujících. V monotónním obvodu chybí hradla NOT, což zajišťuje, že funkce reprezentovaná obvodem je monotónní – zvýšení hodnoty libovolného vstupu nikdy nesníží hodnotu výstupu [7].

2.2.1 P-úplnost MCVP

MCVP je P-úplný problém [7, 5]. Vyhodnocení monotónního obvodu lze provést v polynomiálním čase postupným vyhodnocováním uzlů od vstupů směrem k výstupu. Přestože je problém v třídě P, jeho P-úplnost naznačuje, že pravděpodobně neexistuje efektivní paralelní algoritmus pro jeho řešení – problém vyžaduje sekvenční zpracování.

2.3 Kombinatorické hry na grafu

Kombinatorická hra dvou hráčů na orientovaném grafu je další příklad P-úplného problému [7]. Hra má tyto vlastnosti:

- **Dva hráči:** Každé pole grafu má specifikováno, který hráč je na tahu – Hráč I (první hráč) nebo Hráč II (dvruhý hráč).
- **Konečná pozice:** Hra končí, když je hráč na tahu v pozici bez možných dalších tahů.

Problém spočívá v rozhodnutí, zda Hráč I má výherní strategii ze zadané počáteční pozice:

- **Vstup:** Orientovaný graf (DG), kde každý uzel reprezentuje herní pozici přiřazenou některému z hráčů, hrany reprezentují možné tahy a jeden uzel je označen jako počáteční pozice.
- **Výstup:** Rozhodnutí, zda Hráč I má výherní strategii ze startovní pozice.

Pravidla hry: Hra končí, když se dostane do pozice, kde hráč na tahu nemá žádný možný tah – tento hráč pak prohrává. Hráči se střídají v tazích podle toho, jaký hráč je přiřazen aktuální pozici: je-li uzel přiřazen Hráči I, táhne Hráč I; je-li přiřazen Hráči II, táhne Hráč II.

Výherní strategie: Výherní strategie pro Hráče I je taková posloupnost tahů, která garantuje výhru Hráče I bez ohledu na to, jak hraje Hráč II. Jinými slovy, Hráč I má výherní strategii, pokud existuje způsob, jak vždy volit tahy tak, aby se hra dostala do pozice, kde je Hráč II na tahu a nemá žádný možný tah [7].

2.3.1 P-úplnost kombinatorických her

Tento problém je P-úplný [7, 5]. Určení výherní strategie lze provést v polynomiálním čase pomocí tzv. retrográdní analýzy [7], která zpětně vyhodnocuje pozice od koncových uzlů. I když je problém řešitelný v třídě P, jeho P-úplnost naznačuje, že pravděpodobně neexistuje efektivní paralelní algoritmus – řešení vyžaduje sekvenční zpracování pozic.

2.4 Neprázdnost bezkontextové gramatiky

Problém neprázdnosti jazyka bezkontextové gramatiky (CFG Non-emptiness Problem) je dalším příkladem P-úplného problému [7]. Bezkontextová gramatika představuje formální systém sloužící k generování jazyka.

Formálně je gramatika definována jako čtveřice $G = (N, \Sigma, P, S)$ [8], kde:

- N je konečná množina neterminálních symbolů (neterminálů).
- Σ je konečná množina terminálních symbolů (terminálů), disjunktní s N.
- P je konečná množina přepisovacích pravidel tvaru $A \to \alpha$, kde $A \in N$ a $\alpha \in (N \cup \Sigma)^*$.
- $S \in N$ je počáteční symbol (start symbol).

Problém spočívá v tom, zda daná gramatika generuje alespoň jedno slovo složené pouze z terminálních symbolů.

2.4.1 P-úplnost problému neprázdnosti

Problém neprázdnosti jazyka bezkontextové gramatiky je P-úplný [7, 5]. Tento problém lze vyřešit v polynomiálním čase pomocí algoritmu iterativního označování produktivních neterminálů. I přes řešitelnost v polynomiálním čase je problém P-úplný, což naznačuje obtížnou paralelizovatelnost. Produktivita jednoho neterminálu často závisí na produktivitě jiných neterminálů. Tato vzájemná závislost vyžaduje sekvenční zpracování podobné vyhodnocování MCVP obvodu.

2.5 Principy převodů mezi problémy

Vzájemná převoditelnost P-úplných problémů je klíčovou vlastností, která ilustruje jejich rovnocennost z hlediska výpočetní složitosti.

2.5.1 Převod z MCVP na kombinatorickou hru

V tomto převodu se mapují uzly obvodu MCVP na pozice ve hře dvou hráčů [7]. Každý typ uzlu odpovídá specifické herní situaci:

- **Hradlo OR** vytváří pozici pro Hráče 1. Ten si může vybrat libovolnou následující pozici odpovídající potomkům hradla. Stačí, aby jedna z možností vedla k jeho výhře.
- **Hradlo AND** vytváří pozici pro Hráče 2. Ten vybírá následující pozici a snaží se zabránit výhře Hráče 1. Hráč 1 vyhraje pouze pokud vyhrává ve všech možných pokračováních.
- **Proměnná s hodnotou 1** odpovídá konečné pozici, ve které Hráč 2 nemá žádné tahy – Hráč 1 tedy vyhrává.
- **Proměnná s hodnotou 0** odpovídá konečné pozici, ve které Hráč 1 nemá žádné tahy a prohrává.

2.5.2 Převod z MCVP na bezkontextovou gramatiku

Převod mapuje uzly MCVP obvodu na symboly gramatiky [7]:

- **Kořen obvodu** se mapuje na počáteční symbol gramatiky S.
- **Hradlo AND s potomky** A **a** B vytvoří pravidlo $X \to AB$, kde X reprezentuje hradlo. Řetězec lze z X odvodit právě tehdy, když lze odvodit řetězce z obou potomků A i B.
- **Hradlo OR s potomky** A **a** B vytvoří dvě pravidla: $X \to A$ a $X \to B$.
- **Proměnná s hodnotou 1** generuje epsilon pravidlo $X \to \varepsilon$, což umožňuje odvození prázdného řetězce.
- **Proměnná s hodnotou 0** vytvoří pravidlo $X \to t$, kde t je terminál bez dalších odvozovacích pravidel.

Kapitola 3

Analýza a návrh

V této kapitole se věnujeme popisu technologií, které jsme pro vývoj aplikace zvolili, a detailnímu návrhu architektury celého systému. Zaměřujeme se také na definici klíčových datových struktur a principy návrhu uživatelského rozhraní, které jsou zásadní pro srozumitelnost výukového nástroje.

3.1 Volba technologií

Pro distribuci výukové aplikace jsme zvolili webové technologie, které nabízejí řadu výhod oproti desktopovým nebo mobilním aplikacím. Hlavním argumentem pro toto řešení je maximální dostupnost pro uživatele napříč platformami – aplikace nevyžaduje žádnou instalaci a běží v libovolném moderním prohlížeči. Snadná údržba a okamžitá distribuce aktualizací jsou dalšími přínosy, které webové prostředí poskytuje.

3.1.1 React a Vite

Jako základní kámen vývoje jsme zvolili knihovnu *React* [4]. React nám umožnil rozdělit uživatelské rozhraní na znovupoužitelné komponenty, což výrazně usnadnilo vývoj komplexních modulů pro jednotlivé problémy. Klíčovým konceptem je zde reaktivita – rozhraní se automaticky synchronizuje se stavem dat, což minimalizuje riziko chyb při ruční manipulaci s DOM.

Pro sestavení aplikace jsme využili nástroj *Vite* [11]. Vite nám poskytl rychlé vývojové prostředí díky efektivnímu bundlování a okamžité odezvě při úpravách kódu. Pro finální produkční verzi se Vite stará o optimalizaci souborů, což zajišťuje rychlé načítání aplikace.

3.2 Architektura a struktura kódu

Aplikaci jsme navrhli jako *Single Page Application* (SPA). Veškerá logika a navigace probíhá v rámci jedné stránky, což uživateli poskytuje plynulý zážitek podobný desktopové aplikaci. Kořenem struktury je komponenta `App.jsx`, která spravuje globální stav a řídí přepínání mezi moduly.

Zdrojový kód jsme strukturovali tak, aby byla zajištěna oddělitelnost logiky a vizualizace. Každý modul (MCVP, hry, gramatiky) disponuje vlastní adresářovou strukturou:

- **Hlavní komponenta:** Funguje jako kontejner, který spravuje lokální stav a propojuje vstupní data s algoritmy a vizualizací.
- **Adresář Utils:** Zde jsou soustředěny čisté algoritmy – parsery pro zpracování vstupu, generátory náhodných instancí a evaluátory pro výpočet řešení.
- **Vizualizační komponenty:** Starají se o vykreslení grafů a stromů pomocí knihovny *react-force-graph*.

3.3 Návrh datových struktur

Abychom mohli s P-úplnými problémy efektivně pracovat a vizualizovat je, museli jsme nejprve navrhnout jejich vnitřní reprezentaci v paměti aplikace.

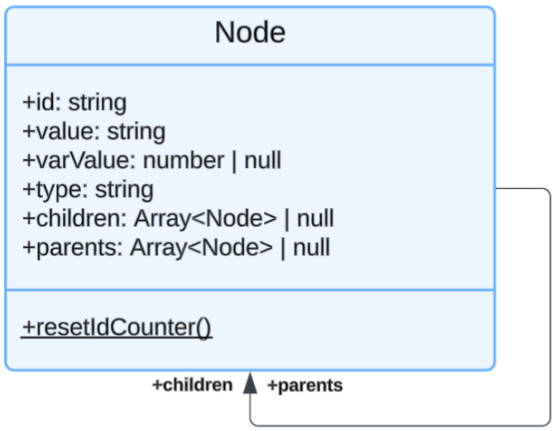
3.3.1 Reprezentace obvodu MCVP

Pro modelování logického obvodu jsme využili orientovaný acyklický graf. Každý uzel obvodu je reprezentován instancí třídy `Node`. Tato třída, jak ukazuje třídní diagram na obrázku 3.1, uchovává informaci o typu operace (AND, OR) nebo hodnotě proměnné. Klíčovou vlastností je seznam odkazů na potomky, což nám umožňuje rekurzivní průchod celým stromem při jeho vyhodnocování.

Obrázek 3.1: Třídní diagram pro hlavní třídy MCVP

3.3.2 Reprezentace herního grafu

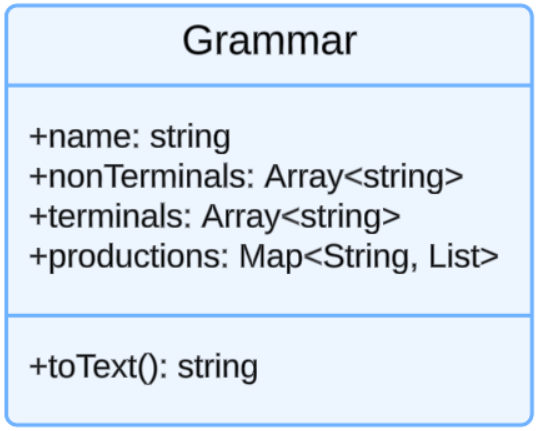
Herní graf kombinatorické hry vyžadoval odlišný přístup, neboť na rozdíl od MCVP může obsahovat cykly. Pro tyto účely jsme navrhli třídy `GamePosition` a `GameGraph` (viz obrázek 3.2). Třída `GamePosition` reprezentuje jednotlivou pozici a uchovává informaci o hráči, který je na tahu, spolu se seznamy předchůdců a následníků, což je nezbytné pro retrográdní analýzu.



Obrázek 3.2: Třídní diagram pro hlavní třídy kombinatorické hry

3.3.3 Reprezentace bezkontextové gramatiky

U bezkontextových gramatik jsme se zaměřili na reprezentaci přepisovacích pravidel. Třída `Grammar` (obrázek 3.3) spravuje množiny symbolů a seznamy produkcí. Každá produkce propojuje levou stranu (neterminál) s polem symbolů na pravé straně. Tento návrh nám dovoluje snadno identifikovat produktivní neterminály v iterativním algoritmu.



Obrázek 3.3: Třídní diagram pro hlavní třídu bezkontextové gramatiky

3.4 Návrh uživatelského rozhraní

Při návrhu rozhraní jsme kladli důraz na to, aby student nebyl zahlcen informacemi a mohl se soustředit na podstatu problému. Rozložení stránky jsme sjednotili napříč všemi moduly:

- **Ovládací panel:** Levá nebo horní část obrazovky, kde uživatel volí způsob zadání vstupu a spouští simulace.
- **Interaktivní plátno:** Hlavní plocha pro vizualizaci grafů, kde uživatel může s prvky přímo manipulovat.
- **Informační panely:** Modální okna nebo postranní panely zobrazující detaily o probíhajícím výpočtu nebo krokovém převodu.

Pro vizualizaci jsme zvolili knihovnu *react-force-graph* [2], která díky fyzikální simulaci automaticky rozmísťuje uzly a hrany, čímž zajišťuje přehlednost i u složitějších instancí. Stylování jsme realizovali pomocí frameworku *Bootstrap 5* [3], který zajistil jednotný vzhled a responzivitu rozhraní na různých velikostech obrazovek.

Kapitola 4

Implementace

V této kapitole detailně popisujeme implementaci jednotlivých modulů aplikace. Zaměřujeme se na technické řešení algoritmů pro parsování, vyhodnocování a generování instancí P-úplných problémů.

4.1 Monotone Circuit Value Problem

Modul pro MCVP tvoří jádro aplikace a slouží jako výchozí bod pro převody na ostatní problémy.

4.1.1 Lexikální a syntaktická analýza

Převod textového výrazu na stromovou strukturu probíhá ve dvou fázích v modulu `Parser.js`:

1. **Tokenizace:** Funkce `tokenize()` rozdělí vstupní řetězec na posloupnost tokenů (závorky, operátory, proměnné) pomocí regulárních výrazů.
2. **Parsování:** Třída `Parser` implementuje rekurzivní sestupný parser, který respektuje prioritu operátorů (OR má nižší prioritu než AND). Výsledkem je hierarchie objektů třídy `Node`.

4.1.2 Algoritmus vyhodnocení

Vyhodnocení obvodu je implementováno v modulu `EvaluateTree.js`. Funkce `evaluateTree()` využívá algoritmus průchodu do hloubky (DFS) s memoizací vyhodnocených uzlů. Tím je zajištěna lineární časová složitost $O(n)$ vzhledem k počtu uzlů. Pro účely výuky jsme implementovali také funkci `evaluateTreeWithSteps()`, která zaznamenává stav obvodu v každém kroku výpočtu.

4.1.3 Interaktivní editace a generování

Kromě textového vstupu jsme vytvořili grafický editor v komponentě `InteractiveMCVPGraph`. Uživatel může přímo v ploše grafu přidávat uzly, měnit jejich typ a vytvářet propojení.

Modul `Generator.js` umožňuje vytvářet náhodné platné DAG struktury. Algoritmus nejprve vygeneruje listy (proměnné) a následně k nim postupně připojuje hradla tak, aby nevznikaly cykly a výsledný graf měl jediný kořen.

4.2 Kombinatorické hry

Implementace kombinatorických her se soustředí na práci s grafy, které mohou obsahovat cykly.

4.2.1 Retrográdní analýza

Jádrem modulu je retrográdní analýza v souboru `ComputeWinner.js`. Na rozdíl od MCVP, kde postupujeme od listů ke kořeni, zde algoritmus začíná u terminálních pozic (uzlů bez následníků) a šíří výsledek (výhra/prohra) zpět proti směru hran. K tomu využíváme frontu (BFS přístup), která zaručuje správné vyhodnocení i v přítomnosti cyklů – pozice, které nelze jednoznačně označit jako výherní nebo prohrávající, zůstávají ve stavu remízy.

4.2.2 Vizualizace a krokování

Pro zobrazení herního grafu využíváme komponentu `DisplayGraph`. Výherní strategie je v grafu zvýrazněna odlišnou barvou hran. Krokové vyhodnocení v komponentě `StepByStepGame` umožňuje uživateli sledovat, jak se stavy prohry a výhry postupně propagují grafem.

4.3 Bezkontextové gramatiky

Modul gramatik řeší problém neprázdnosti jazyka pomocí identifikace produktivních neterminálů.

4.3.1 Identifikace produktivních symbolů

Algoritmus v modulu `GrammarEvaluator.js` pracuje iterativně. V každém kroku označí neterminály, pro které existuje pravidlo s pravou stranou složenou výhradně z terminálů nebo již označených produktivních neterminálů. Výpočet končí, jakmile se množina produktivních symbolů v dané iteraci nezmění.

4.3.2 Rekonstrukce derivačního stromu

Pokud je počáteční symbol označen jako produktivní, aplikace dokáže rekonstruovat konkrétní derivaci jako důkaz neprázdnosti. Tento proces probíhá rekurzivně – pro každý neterminál náhodně vybereme jedno z jeho produktivních pravidel a budujeme strom až k terminálním symbolům. Abychom předešli nekonečným derivacím u rekurzivních gramatik, omezili jsme hloubku stromu a implementovali preferenci nerekurzivních pravidel při překročení určité meze.

4.4 Implementace převodů mezi problémy

Klíčovou funkcí aplikace je vizualizace redukcí mezi P-úplnými problémy.

4.4.1 Redukce na kombinatorickou hru

Převod implementuje třída `MCVPToGameStepGenerator`. Prochází MCVP graf a pro každý uzel generuje odpovídající herní pozici: OR hradla se mění na pozice Hráče I, AND hradla na pozice Hráče II. Terminální hodnoty 0 a 1 jsou převedeny na prohrávající pozice pro dané hráče. Implementace využívá memoizaci, aby každý uzel MCVP odpovídal právě jedné pozici v herním grafu.

4.4.2 Redukce na bezkontextovou gramatiku

Převod zajišťuje komponenta `MCVPtoGrammarConverter`. Algoritmus rekurzivně mapuje strukturu obvodu na přepisovací pravidla. AND hradla vytvářejí pravidla s více symboly na pravé straně (konjunkce), zatímco OR hradla vytvářejí alternativní pravidla pro tentýž neterminál (disjunkce). Epsilon pravidla jsou využita k reprezentaci logické hodnoty 1.

Kapitola 5

Závěr

Cílem této práce bylo vytvořit interaktivní webovou aplikaci pro vizualizaci a demonstraci převoditelnosti a řešení P-úplných problémů na příkladu tří vybraných P-úplných problémů: Monotone Circuit Value Problem, kombinatorických her na grafu a problému neprázdnosti jazyka bezkontextové gramatiky.

5.1 Dosažené výsledky

Výsledná aplikace splňuje všechny stanovené cíle a poskytuje jednotné prostředí pro práci se třemi P-úplnými problémy. Pro každý problém je implementováno kompletní řešení zahrnující:

- **Flexibilní vstupní systém:** Uživatelé mohou zadávat instance problémů třemi způsoby – manuálním zadáním, generováním náhodných instancí nebo pomocí předpřipravených příkladů. Tyto možnosti umožňují jak experimentování s vlastními příklady, tak rychlé testování algoritmu na náhodných datech.
- **Vizualizaci řešení:** Každý problém je doprovázen grafickou reprezentací, která využívá knihovnu *react-force-graph-2d* pro interaktivní zobrazení grafových struktur. Uživatel může manipulovat se zobrazenými grafy, přibližovat je a přesouvat uzly pro lepší orientaci.
- **Krokové vyhodnocení:** Implementace krokovatelného průchodu algoritmů umožňuje sledovat každý jednotlivý krok výpočtu s textovým vysvětlením.
- **Export a import dat:** Možnost ukládání a načítání instancí problémů ve formátu JSON podporuje sdílení příkladů a vytváření knihoven testovacích případů.

Aplikace taktéž demonstruje dva konkrétní převody z MCVP na kombinatorickou hru a na bezkontextovou gramatiku. Oba převody jsou implementovány krokovatelně, takže uživatel může sledovat, jak se jednotlivé uzly obvodu transformují na odpovídající struktury v cílovém problému.

Tato vizualizace názorně ilustruje techniku polynomiálních redukcí a vzájemnou převoditelnost P-úplných problémů.

5.2 Vzdělávací přínos

Aplikace představuje vzdělávací nástroj pro výuku teorie složitosti s interaktivním přístupem. Krokové vyhodnocení s textovými vysvětleními umožňuje pochopit fungování algoritmů na konkrétních příkladech, zatímco vizualizace převodů demonstruje vzájemnou převoditelnost P-úplných problémů. Generování náhodných instancí pak podporuje experimentální učení a pozorování chování algoritmů na různých strukturách vstupů.

5.3 Možnosti dalšího rozvoje

Přestože aplikace poskytuje funkční implementaci všech plánovaných funkcí, existuje prostor pro další rozšíření:

- **Další P-úplné problémy:** Aplikace by mohla být rozšířena o další P-úplné problémy, jako je například vyhodnocování booleovských formulí v konjunktivní normální formě nebo problém dosažitelnosti v grafech s omezenou šířkou.
- **Více převodů:** Implementace dalších směrů převodů – například z kombinatorických her na gramatiky nebo opačným směrem z gramatik na MCVP – by poskytla kompletnější obraz vzájemné převoditelnosti těchto problémů.
- **Výkonnostní optimalizace:** Pro velmi velké instance (stovky uzlů) by mohly být implementovány optimalizace vykreslování a výpočtu, například pomocí virtualizace zobrazení nebo progresivního vykreslování.

Výsledná aplikace může sloužit jako doplněk k tradičním výukovým materiálům v kurzech teorie složitosti a teoretické informatiky, kde pomáhá studentům lépe pochopit abstraktní koncepty prostřednictvím konkrétních interaktivních příkladů.

Literatura

1. ARORA, Sanjeev; BARAK, Boaz. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. ISBN 978-0521424264.
2. ASTURIANO, Vasco. *react-force-graph: React component for 2D, 3D, VR and AR force directed graphs [online]* [GitHub]. 2024. Dostupné také z: `https://github.com/vasturiano/react-force-graph`. [cit. 2026-02-03].
3. BOOTSTRAP TEAM. *Bootstrap: Build fast, responsive sites [online]* [OpenJS Foundation]. 2024. Dostupné také z: `https://getbootstrap.com/`. Verze 5.3.8. [cit. 2026-02-03].
4. META PLATFORMS, INC. *React: The library for web and native user interfaces [online]* [Meta Open Source]. 2024. Dostupné také z: `https://react.dev/`. Verze 19.2. [cit. 2026-02-03].
5. MIYANO, Satoru; SHIRAISHI, Shunsuke; SHOUDAI, Takayoshi. *A List of P-Complete Problems [online]*. 1990-12. Tech. zpr., RIFIS-TR-CS-17. Fukuoka: Kyushu University, Research Institute of Fundamental Information Science. Dostupné také z: `https://catalog.lib.kyushu-u.ac.jp/opac_download_md/3123/rifis-tr-17.pdf`. [cit. 2026-02-03].
6. PAPADIMITRIOU, Christos H. *Computational Complexity*. Addison Wesley, 1994. ISBN 978-0201530827.
7. SAWA, Zbyněk. *Teoretická informatika – Podklady pro přednášky [online]* [Ostrava: VŠB-TUO, Katedra informatiky]. 2025. Dostupné také z: `https://www.cs.vsb.cz/sawa/ti/slides/ti-slides-08.pdf`. [cit. 2026-02-03].
8. SAWA, Zbyněk. *Úvod do teoretické informatiky – Bezkontextové gramatiky [online]* [Ostrava: VŠB-TUO, Katedra informatiky]. 2014. Dostupné také z: `https://www.cs.vsb.cz/kot/download/uti2014/uti-10-cz.pdf`. [cit. 2026-02-03].
9. SAWA, Zbyněk. *Úvod do teoretické informatiky – Turingovy stroje [online]* [Ostrava: VŠB-TUO, Katedra informatiky]. 2014. Dostupné také z: `https://www.cs.vsb.cz/kot/download/uti2014/uti-08-cz.pdf`. [cit. 2026-02-03].

10. SAWA, Zbyněk. *Úvod do teoretické informatiky – Výpočetní složitost algoritmů [online]* [Ostrava: VŠB-TUO, Katedra informatiky]. 2014. Dostupné také z: `https://www.cs.vsb.cz/kot/download/uti2014/uti-12-cz.pdf`. [cit. 2026-02-03].

11. VITE TEAM. *Vite: The Build Tool for the Web [online]* [VoidZero Inc.]. 2024. Dostupné také z: `https://vite.dev/`. Verze 6. [cit. 2026-02-03].