Západočeská univerzita v Plzni

Fakulta pedagogická

Katedra výpočetní a didaktické techniky

Počítačové metody eliminace kvantifikátorů v *R* a možnosti jejich využití

Mgr. Lukáš HONZÍK

Specializace v pedagogice,

obor Informační a komunikační technologie ve vzdělávání

Vedoucí práce: doc. RNDr. Jaroslav Hora, CSc.

Plzeň, 2012

Prohlašuji, že jsem disertační práci vypracoval sám s použitím uvedené literatury a zdrojů informací.

Plzeň, 1. listopadu 2012

.....

Děkuji za odbornou pomoc, vedení a řadu cenných podnětů mému školiteli doc. RNDr. Jaroslavu Horovi, CSc.
Zároveň bych rád poděkoval mé rodině a přátelům za jejich podporu a trpělivost a v neposlední řadě také Mgr. Ondřeji Davídkovi, bez jehož pomoci by mé začátky v QEPCADu byly více než složité.

Obsah

Anotace: Eliminace kvantifikátorů v oboru reálných čísel je jednou z mladších oblastí na pomezí matematiky, logiky a výpočetní techniky. Pomocí metody nazývané cylindrická algebraická dekompozice dovoluje zjednodušit matematické kvantifikované formule do jednoduššího tvaru neobsahujícího kvantifikátory. Jelikož se jedná o vcelku složitý postup, jsou velmi užitečné programy počítačové algebry (například program Mathematica od firmy Wolfram Research). Mnoho matematických úloh (jako třeba řešení rovnic a nerovnic) může být převedeno právě na matematické formule obsahující kvantifikátory, u nichž pak eliminace dovoluje zjistit řešení nebo určit, zda jsou řešitelné. Z toho důvodu může být tato metoda přínosem nejen pro matematiky samotné, ale též pro učitele matematiky a talentované žáky.
Klíčová slova: eliminace kvantifikátorů, CAD, cylindrická algebraická dekompozice, Alfred Tarski, George E. Collins, počítačová algebra, Mathematica, trojúhelníková dekompozice, regulární řetězce

Annotation: Quantifier elimination over real fields is a discipline connected with mathematics, logic and computer science. Using so called cylindrical algebraic decomposition it allows to simplify mathematical formulas with quantifiers into quantifier-free formulas. Since this is quite complex problem programs of computer algebra (such as Mathematica by Wolfram Research) are very helpful. Many mathematical problems (for example equations and inequations) can be transformed into quantified formulas and the elimination is a way to solve them or find out if they are solvable. Therefore this method may be useful not only for mathematicians but for math teachers and talented pupils too.
Key words: quantifier elimination, CAD, cylindrical algebraic decomposition, Alfred Tarski, George E. Collins, computer algebra, Mathematica, triangular decomposition, regular chains

1 Úvod

Eliminace kvantifikátorů je poměrně mladou problematikou na pomezí matematiky, logiky a v neposlední řadě i výpočetní techniky a její počátky nalezneme v první polovině 20. století. Tehdy se jí zabýval významný polský matematik Alfred Tarski, který však i přes nemalé úspěchy nepočítal s tím, že by se – zvláště kvůli své nadměrné složitosti – dala vhodným a reálným způsobem využívat, a sám nakonec prohlásil, že zkoumání tohoto problému bude nutné ukončit, jelikož nepřinese žádné cenné využitelné výsledky.

Zlom pak nastal až ve druhé polovině dvacátého století s příchodem výkonnější výpočetní techniky a objevením metody cylindrické algebraické dekompozice (CAD, Cylindrical Algebraic Decomposition), již představil světu matematik George E. Collins. Tato metoda dovoluje únosnou algoritmizaci celého postupu.

Porovnáme-li obě zmíněné metody s ohledem na časovou složitost, je zřejmé, že metoda CAD je schůdnější. V roce 1974 ukázali počítačoví odborníci Michael O. Rabin a Michael J. Fischer, že složitost Tarského přístupu není omezena žádnou „věží z exponentů“ $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$, zatímco Collinsova metoda umožnila snížit časovou složitost na 2^{2^n}, tento algoritmus je tedy v nejhorším případě „dvojnásobně exponenciální“. I to může být mnoho, pokud bychom uvažovali úlohy, kdy je počet n proměnných obsažených ve formuli velký. V úlohách školské matematiky však bývá jejich počat omezený. [BEE03]

Později přibyl ještě další přístup stavějící na regulárních řetězcích a trojúhelníkové dekompozici, byť v případě tohoto přístupu je celý daný problém převeden na řešení polynomiálních soustav rovnic a nerovnic s parametry, přičemž výsledek se dá interpretovat jako matematická formule bez kvantifikátorů ekvivalentní s formulí původní. Pro bližší informace odkážeme čtenáře na práci *Solving Polynomial Systems via Triangular Decomposition* autorů Changbo Chena a Marca Moreno Maza. [CHE11]

Přestože se na první pohled může zdát, že se jedná o „těžkou“ vědeckou matematiku, setkávají se s ní v jejím jednodušším pojetí již studenti na střední škole, aniž by se tento fakt od svých kantorů dověděli. Od studentů je například při výkladu a cvičení učiva o kvadratických rovnicích běžně vyžadována odpověď na otázku, kdy má normovaná kvadratická rovnice $x^2 + px + q = 0$ reálný kořen. Očekávají, že odpověď bude: právě tehdy, když diskriminant D je nezáporný, čili $p^2 - 4q \geq 0$.

Zapišme vše pomocí kvantifikátorů

$$(\exists\, x)\colon x^2 + px + q = 0 \Leftrightarrow p^2 - 4q \geq 0.$$

Na levé straně je zapsána formule s existenčním kvantifikátorem $\exists$ obsahující jím vázanou proměnnou x. Vpravo je ekvivalentní formule, která již neobsahuje kvantifikátor ani vázanou proměnnou, tak byla provedena eliminace kvantifikátoru.

Na zástupce některých dalších nepříliš složitých úloh dostupných již na SŠ narazíme ve 3. kapitole této práce. Při jejich řešení si však vesměs vystačíme s „lidskými“ přístupy, které pro vyřešení těchto typů příkladů většinou dostačují.

Avšak v případě složitějších úloh (například v případě výskytu více polynomiálních nerovnic, vyššího množství proměnných v zadané matematické formuli, či vysokých mocnin proměnných) tyto metody selhávají, případně se stávají poměrně těžkopádnými a na řadu musí přijít matematické programy.

Jako motivační příklad pro tuto situaci použijme úlohu vycházející z článku *Cylindrical Algebraic Decomposition I: The Basic Algorithm* autorů Dennise S. Arnona, George E. Collinse a Scotta McCalluma vydaného v roce 1982 v *SIAM Journal on Computing*.

Mějme kvantifikovanou formuli

$$(\exists y)\colon y^4 - 2y^3 + y^2 - 3x^2y + 2x^4 = 0$$

a využijme výpočetní prostředí Wolfram Mathematica jako zástupce programů počítačové algebry, které eliminaci kvantifikátorů zvládají, k jejímu zjednodušení. Zadáním příkazu **Resolve [Exists [y, y^4 – 2y^3 + y^2 – 3x^2y + 2x^4 == 0]]** počítači uložíme tuto formuli zjednodušit do tvaru bez kvantifikátorů. Ve velmi krátké době obdržíme výsledek `True`, který interpretujeme tak, že skutečně existuje nějaká (alespoň jedna) reálná hodnota proměnné y, pro niž je splněna zadaná rovnost. Použijeme-li navíc příkaz **Timing [výraz]**, jehož výstupem je kromě samotného výsledku i čas potřebný k provedení celého výpočtu (příkaz tedy bude ve tvaru **Timing [Resolve [Exists [y, y^4 – 2y^3 + y^2 – 3x^2y + 2x^4 == 0]]]**), dostaneme odpověď `{0.,True}`, podle které trvalo zjištění pravdivosti zadané formule skutečně krátký okamžik (v tomto případě tak krátký, že jej program ve svém výpisu ohodnotil dobou trvání 0 sekund).

Zkusíme-li naproti tomu počítači k řešení zadat trochu pozměněnou matematickou formuli

$$(\forall y)\colon y^4 - 2y^3 + y^2 - 3x^2y + 2x^4 = 0,$$

bude trvat zjištění její pravdivosti o něco déle. Na příkaz **Timing [Resolve [ForAll [y, y^4 – 2y^3 + y^2 – 3x^2y + 2x^4 == 0]]]** obdržíme odpověď `{0.16,False}`. Jak je vidět, tentokrát již výpočet, kterým počítač zjistil, že zadané tvrzení je nepravdivé, zabral času více, nicméně stále se jedná o relativně krátkou dobu.

Budeme-li tyto pokusy opakovat, je velmi pravděpodobné, že se další výsledné časy budou navzájem lišit. Přirozeně záleží na počtu spuštěných operací, na velikosti a zaplnění operační paměti a dalších faktorech, které rychlost výpočtu ovlivňují. I přes to si však lze utvořit představu o tom, jak rychle celá operace proběhne.

Nejen zvídavé žáky a studenty, kteří s problematikou počítačového dokazování přijdou do kontaktu, pak přirozeně může napadnout otázka, jak počítač, potažmo matematický program, k výsledku vlastně došel a jak eliminaci kvantifikátorů v zadaném tvrzení provedl. Odpovědí na tuto záhadu jsou právě do programů počítačové algebry implementované metody eliminace kvantifikátorů a především možnost jejich algoritmizace.

V této souvislosti je nutné poznamenat, že v současné době je téma aktuální a stále v této oblasti matematiky probíhá další studium a optimalizace procesů, nicméně přeci jen jde o poměrně úzké zaměření, které však může poskytnout podporu a rozšíření znalostí nejen studentům na středních a vysokých školách, kteří se věnují matematice a výpočetní technice, ale též učitelům, již tyto studenty vzdělávají a vedou.

Zároveň je také přirozené se ptát, zda lze eliminaci kvantifikátorů provádět jen v určitých jednoduchých formulích podobných výše zmíněným nebo zda je realizovatelná obecně. To je záležitost matematické logiky a daný problém vyřešil právě Alfred Tarski. Naprosto korektní výklad všech náležitostí je proveden ve specializovaných textech, například v češtině v publikaci Vítězslava Švejdara *Logika: Neúplnost, složitost a nutnost* (Praha: Academia, 2002).

Cílem této disertační práce je pokusit se čtenáři srozumitelně předložit hlavní ideu metody cylindrické algebraické dekompozice, která může být v procesu eliminace kvantifikátorů používána, a též její automatizování pomocí matematického aparátu, tento výklad pak doplnit vhodnými ukázkovými příklady jednodušších i složitějších ilustračních úloh a připravit tak soubor úloh na eliminaci kvantifikátorů použitelnou jako metodický základ pro učitele. Jak uvidíme dále v textu, je zde i mnoho souvislostí s historií matematiky a výpočetní techniky.

Z tohoto důvodu přenecháme korektní výklad záležitostí v matematické logiky odborným textům a budeme prostě hovořit o tom, že eliminaci kvantifikátorů

provádíme v tělese reálných čísel, i když by bylo korektní říci, že je prováděna v elementární teorii formálně reálně uzavřených těles. Při tom je ale důležité si alespoň uvědomit, že kvantifikovány mohou být jen prvky tělesa *R*, tj. samotná reálná čísla, nikoliv jejich množiny, jako kupříkladu intervaly, množina celých čísel atd. Nemůžeme také konstruovat funkce a pracovat s nimi. Na druhé straně při konstrukci formulí lze pracovat nejen s rovnostmi, ale i se všemi druhy nerovností. Můžeme tedy vytvářet formule v proměnných *x, y, z, ...* pomocí operací sčítání (odčítání) a násobení, jsou „povoleny“ celočíselné koeficienty (vzniknout z konstant 0 a 1) a smíme používat symboly $<$ a $=$, a tedy i $>$, $\leq$, $\geq$. Prakticky to znamená, že můžeme zapisovat polynomiální rovnice a nerovnice a z nich vytvářet logické kombinace pomocí logických spojek. Proměnné obsažené v těchto formulích pak lze kvantifikovat.

Práce je rozdělena do pěti hlavních kapitol. První z nich je věnována historickému exkurzu a historickým souvislostem, jakož i vývoji studované problematiky. Krátce je zmíněno zavádění automatů a později počítačů do výuky a vzdělávání (nejen matematiky), na což je navázáno shrnutím života polského matematika a logika Alfreda Tarského a zmíněním přínosu jeho i jeho studentů a kolegů k aplikovatelnosti a řešitelnosti problematiky eliminace kvantifikátorů.

V další kapitole jsou čtenáři připomenuty některé jednodušší příklady a úlohy středoškolské matematiky, jejichž řešení je s tématem eliminace kvantifikátorů pevně svázáno, byť toto je prozatím ponecháno v rovině řešení logickou úvahou a metodou nulových bodů.

Následuje kapitola, ve které je čtenáři předložena hlavní idea cylindrické algebraické dekompozice, metody rozpracované Georgem E. Collinsem a vylepšované jeho studenty, která je vhodně použitelná i pro složitější úlohy, než jaké byly zmíněny v kapitole předchozí. Teorie je zde vysvětlena na konkrétní úloze a doplněna grafickými znázorněními dané situace pro její lehčí pochopení.

V další části práce jsou tyto teoretické myšlenky aplikovány a s využitím matematického aparátu algoritmizovány. Čtenáři se tak dozvědí, co vše stojí v pozadí cylindrické algebraické dekompozice a na čem jsou založeny možnosti a schopnosti matematických programů při řešení zmíněných úloh. Pro pořádek je kapitola rozdělena do několika částí, v nichž jsou postupně probrány všechny jednotlivé fáze algoritmu cylindrické algebraické dekompozice doplněné příslušnými ilustračními příklady. V přílohách práce se pak nachází vývojovým diagramem vyjádřený algoritmus jak

samotné cylindrické algebraické dekompozice, tak její včlenění do celé algoritmizované eliminace kvantifikátorů.

Zároveň jsou v této kapitole zmíněny i možnosti vylepšení celého algoritmu, například zavedením tzv. Hongova a McCallumova operátoru, užitím částečné cylindrické algebraické dekompozice, či různými metodami konstrukce výstupních formulí.

Nezanedbatelnou částí práce je pak poslední, ve své podstatě praktická kapitola věnovaná práci v programech počítačové algebry a řešení příkladů tamtéž. Jmenovitě se jedná o programy Wolfram Mathematica, Maple a aplikaci QEPCAD, jenž je pro eliminaci kvantifikátorů přímo určena. Čtenář je krátce seznámen s každým programem, je řečeno několik slov k jejich vývoji a dostupnosti, což je následně doplněno ukázkami práce s příslušnými příkazy, s nimiž uživatel přijde do styku. Vše je nakonec završeno souborem příkladů řešených v těchto konkrétních programech, takže si čtenář může utvořit představu o jejich možnostech, o způsobu přístupu k jejich řešení a též je mezi sebou porovnat.

Přestože je předložená práce obhajována v oblasti výpočetní techniky, jsou v jejím závěru zdůrazněny některé významné příspěvky pro matematiku a její (vysokoškolskou) výuku (vícerozměrná integrace, počítačové důkazy vět) a je zde uveden i jeden strojový důkaz obtížné nerovnosti.

2 Systém elementární algebry

Moderní matematická logika hovoří o elementární teorii formálně reálně uzavřených těles. V této práci zavedeme potřebné pojmy ve stylu původních Tarského prací a vyhneme se tak přílišnému zaměření na oblast matematické logiky.

V dalším textu se budeme věnovat nejen eliminaci kvantifikátorů v tělese reálných čísel, ale obecně práci s matematickými formulemi, rovnicemi a nerovnicemi. Z tohoto důvodu je tedy vhodné formálním způsobem popsat, resp. čtenáři připomenout, systém elementární algebry. Proto nejprve v této kapitole zopakujeme a ujasníme znalosti týkající se právě této oblasti matematiky.

Pojmem proměnná budeme v dalším textu rozumět některý nebo některé z následujících symbolů:

$$x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots,$$

přičemž těchto proměnných je nekonečně mnoho a intuitivně je chápeme jako způsob symbolické reprezentace objektů z dané množiny. Tu nazýváme obor proměnné nebo také definiční obor, v našem případě proměnné mohou zastupovat libovolné prvky z množiny reálných čísel.

Dále zaveďme pojem algebraické konstanty a znaménka algebraických operací. Pod pojmem algebraická konstanta rozumíme jeden ze symbolů

$$\ldots, -2, -1, 0, 1, 2, \ldots,$$

znaménka algebraických operací označujeme

$$+, \cdot,$$

přičemž znaménko + značí sčítání a znaménko · násobení.

2.1 Algebraické členy a atomické formule

Nyní máme k dispozici všechny pojmy nutné k zavedení pojmu algebraického členu. Algebraický člen (nebo také výraz) chápeme jako libovolné smysluplné vyjádření sestavené z proměnných a algebraických konstant spojených do jednoho celku prostřednictvím algebraických znamének. Například zápisy

$$x, x + y + 4, [x_1 + x_2]\cdot 2\cdot x_3$$

jsou algebraickými členy, zatímco kupříkladu zápisy

$$x +, x\cdot\sqrt{3}$$

nikoliv, neboť v prvním případě se nejedná o smysluplné vyjádření, ve druhém případě zápis obsahuje konstantu $\sqrt{3}$, která podle předchozího nepatří mezi proměnné ani mezi konstanty.

Pro jednodušší vyjadřování a zápisy výrazů zaveďme symboliku pro konečné součty a součiny, kde pro libovolnou posloupnost algebraických členů α_1, α_2, ... platí rekurentní předpis

$$\sum_{i=1}^{1} \alpha_i = \alpha_1, \ \sum_{i=1}^{k+1} \alpha_i = \sum_{i=1}^{k} \alpha_i + \alpha_{k+1}$$

pro sčítání a

$$\prod_{i=1}^{1} \alpha_i = \alpha_1, \ \prod_{i=1}^{k+1} \alpha_i = \prod_{i=1}^{k} \alpha_i \cdot \alpha_{k+1}$$

pro násobení. Potom platí

$$\sum_{i=1}^{n} \alpha_i = \alpha_1 + \alpha_2 + \ldots + \alpha_n, \ \prod_{i=1}^{n} \alpha_i = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n.$$

Pokud navíc máme speciálně $\alpha_1 = \alpha_2 = \ldots = \alpha_n = \alpha$, můžeme psát

$$\sum_{i=1}^{n} \alpha_i = n \cdot \alpha, \ \prod_{i=1}^{n} \alpha_i = \alpha^n.$$

Pro vyjádření tzv. atomických formulí použijeme relační symboly

$$<, =, >,$$

přičemž atomické formule značíme *A* a rozumíme jimi vyjádření za použití relačních symbolů ve tvaru

$$\alpha < \beta, \ \alpha = \beta, \ \alpha > \beta,$$

kde α, β jsou libovolné algebraické výrazy.

První a třetí zápis jsou nerovnostmi, druhý zápis je nazýván rovností. U atomických formulí jde v závislosti na hodnotě dosazené do proměnné z příslušného oboru proměnné rozhodnout o její pravdivosti či nepravdivosti.

Atomické formule můžeme dále rozšiřovat a spojovat do složitějších celků matematických formulí prostřednictvím logických spojek

$$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow.$$

První symbol $\neg$ reprezentuje unární logická operace negace[1], symboly $\wedge$, $\vee$ reprezentují *n*-ární logickou operaci konjunkce a disjunkce a konečně značky $\Rightarrow$, $\Leftrightarrow$ reprezentují binární logickou operaci implikace a ekvivalence.

Podle zákonů platících pro zmíněné logické operace postupně řekneme:

a) Je-li Φ matematická formule, pak formule $\neg\Phi$ je pravdivá právě tehdy, když Φ je nepravdivá.

b) Jsou-li Φ a Θ matematické formule, pak formule $(\Phi \wedge \Theta)$ je pravdivá právě tehdy, když jsou pravdivé obě formule Φ a Θ zároveň, a formule $(\Phi \vee \Theta)$ je pravdivá právě tehdy, když je pravdivá alespoň jedna z formulí Φ nebo Θ.

c) Jsou-li Φ a Θ matematické formule, pak formule $\Phi \Rightarrow \Theta$ je pravdivá právě tehdy, když Φ není pravdivá nebo Θ je pravdivá.

d) Jsou-li Φ a Θ matematické formule, pak formule $\Phi \Leftrightarrow \Theta$ je pravdivá právě tehdy, když obě formule Φ a Θ jsou buď pravdivé anebo nepravdivé.

Poznámka: Pro naši další práci bude také vhodné uvést i operaci odčítání, která je značena znaménkem –, a pro libovolné dva algebraické členy α, β platí rovnost

$$\alpha - \beta = \alpha + (-1 \cdot \beta). \blacklozenge$$

Další podrobnější informace lze nalézt nejen v [TAR57].

2.2 Kvantifikátory a matematické formule

V práci se dále budeme setkávat se symboly – kvantifikátory dvou základních druhů

$$\exists, \forall,$$

první z nich je nazýván existenční (nebo také malý) kvantifikátor, druhý je označován jako obecný (či univerzální, velký).

Oba kvantifikátory jsou ve formulích zapisovány spolu s takzvanými kvantifikovanými proměnnými, případně s jejich uspořádanými *n*-ticemi, a to například v následujících tvarech

$$\exists x, \forall x, (\exists x)(\forall y), \forall x, y, z, ...$$

Tento zápis je pak zpravidla doprovázen nějakou vlastností *V*, která pro kvantifikované a případně i volné prvky platí.

---

[1] V české odborné literatuře se také kromě značení $\neg p$, kde p je atomická formule, můžeme setkat s označením p' nebo $\overline{p}$. V angloamerických literárních i elektronických zdrojích se naopak užívá zápisu $\sim p$, s kterýmžto se čtenář setká v kapitole 7.2 o programu QEPCAD.

Kvantifikátor s kvantifikovanou proměnnou x a vlastností $V(x)$, píšeme $(\exists x)$: $V(x)$, čteme jako „existuje x takové, pro které platí vlastnost $V(x)$“ a říká, že v dané množině[2] je alespoň jeden prvek takový, pro nějž platí zapsaná vlastnost $V(x)$. Kupříkladu kvantifikovanou formuli

$$(\exists x)\colon x^2 = 0$$

čteme: „Existuje x (reálné) takové, že jeho druhá mocnina je rovna 0.“

Naproti tomu obecný kvantifikátor s proměnnou x a vlastností $V(x)$, píšeme $(\forall x)$: $V(x)$, budeme číst jako „pro všechna x platí vlastnost $V(x)$“ a chápeme jej tak, že daná vlastnost $V(x)$ platí pro všechny prvky množiny bez výjimky. Například formuli

$$(\forall x)\colon x < 5$$

přečteme: „Pro všechna x (reálná) platí, že jejich hodnota je menší než číslo 5.“

Kvantifikátory mohou být mezi sebou i různým způsobem kombinovány, jako ukázku můžeme vzít třeba zápis

$$(\exists K, K > 0)(\forall x)\colon |f(x)| < K,$$

který přečteme: „Existuje kladné (reálné) K takové, že pro všechna x platí, že absolutní hodnota funkční hodnoty $f(x)$ je menší než číslo K.“ Pro úplnost dodejme, že takto pomocí kvantifikované formule lze zavést omezenost funkce $f(x)$.

Kromě toho se ještě zmiňme o dalších dvou verzích existenčního kvantifikátoru, a sice $\exists!$ a $\exists_n$. První z nich spolu s proměnnou x a vlastností $V(x)$ $(\exists! x)$: $V(x)$ čteme jako “existuje právě jedno x, pro které platí vlastnost $V(x)$” a pro takový zápis pak platí, že v dané množině existuje takový prvek x s vlastností $V(x)$ právě jeden (jinými slovy můžeme situaci vnímat tak, že takový prvek x s vlastností $V(x)$ je alespoň jeden a zároveň maximálně jeden). Druhá verze existenčního kvantifikátoru $(\exists_n x)$: $V(x)$ pak přečteme jako „existuje právě n prvků x, pro něž platí vlastnost $V(x)$“. Za speciální typ posledně jmenované verze existenčního kvantifikátoru by se dal označit kvantifikátor $\overset{\infty}{\exists}$. Zápis $(\overset{\infty}{\exists} x)$: $V(x)$ pak přečteme „existuje nekonečně mnoho x, pro něž platí $V(x)$“ a značí, že takových prvků x, pro které vlastnost $V(x)$ platí, je nekonečně mnoho, avšak nemusí se jednat o všechny prvky dané množiny. Jako příklad můžeme uvést třeba $(\overset{\infty}{\exists} x)$: $2|x$, čili „existuje nekonečně mnoho sudých čísel“ (ovšem ne všechna čísla jsou sudá).

---

[2] Není-li uvedeno jinak, jde většinou o množinu reálných čísel, avšak zápis může být upřesněn v tom smyslu, že daná proměnná je prvkem jiné množiny.

Poznámka: Zajímavý je také vztah mezi oběma kvantifikátory, vhodnou úpravou v zápisu formule spolu s použitím symbolu negace $\neg$ je totiž možné jeden kvantifikátor vyjádřit pomocí druhého, přičemž obě tyto formule budou navzájem ekvivalentní. Formuli tvaru $(\forall x)$: $V(x)$ takto lze přepsat do tvaru $\neg\exists x$: $\neg V(x)$, čili místo „pro všechna x platí vlastnost $V(x)$“ zapíšeme „neexistuje x takové, že pro něj neplatí vlastnost $V(x)$“. Naopak formuli $(\exists x)$: $V(x)$ přepíšeme za použití obecného kvantifikátoru do tvaru $\neg\forall x$: $\neg V(x)$ a můžeme číst „není pravda, že pro všechna x neplatí vlastnost $V(x)$“. ♦

Zatím jsme se zabývali kvantifikátory a spolu s nimi kvantifikovanými proměnnými. V matematických formulích se ale mohou vyskytnout i proměnné nekvantifikované, tedy tzv. volné, které je nejen pro naši další práci vhodné odlišit právě od proměnných kvantifikovaných.

Neformálně jde říci, že volná proměnná x je taková ve formuli se vyskytující proměnná, která není v matematické formuli svázána (spojena) s žádným kvantifikátorem. Formální definici je možné zavést následujícím způsobem.

Definice 2.2.1: Nechť A je atomická formule, potom proměnná x je volná právě tehdy, když se vyskytuje v této atomické formuli A. Dále:

a) Proměnná x je volná ve formuli $(\exists y)$: Φ, resp. ve formuli $(\forall y)$: Φ právě tehdy, když x je volná ve formuli Φ a zároveň $x \neq y$.

b) Proměnná x je volná ve formuli $\neg\Phi$ právě tehdy, když x je volná ve formuli Φ.

c) Proměnná x je volná ve formuli $(\Phi \wedge \Theta)$ a ve formuli $(\Phi \vee \Theta)$ právě tehdy, když je volná ve formuli Φ a zároveň i ve formuli Θ. ♦

Z toho důvodu řekneme, že proměnná x je volná ve formulích

$$x = 1,$$

$$(\exists y)\colon y = x^2 - 1,$$

$$(\forall y)(\exists a)\colon (y = a \cdot x) \wedge (x \neq 0).$$[3]

Naproti tomu v následujících případech proměnná x není volnou proměnnou.

$$y = 1,$$

$$(\forall x)\colon x^2 \geq 0.$$[4]

[3] V prvním případě jde o samotnou atomickou formuli, ve které se vyskytuje proměnná x, ve druhém i třetím případě se sice vyskytují vázané proměnné y a a, avšak platí $y \neq x \neq a$.

Na závěr ještě dodejme dvě následující poznámky.

Poznámka: Formule obsahující logické spojky $\Rightarrow$ a $\Leftrightarrow$ lze vcelku jednoduše přepsat pomocí vyjádření obsahujícího zbylé tři logické spojky $\neg$, $\wedge$ a $\vee$.

Proto jsou-li Φ a Θ formule, pak implikaci

$$\Phi \Rightarrow \Theta$$

můžeme vyjádřit též jako formuli tvaru

$$\neg\Phi \vee \Theta$$

a ekvivalenci

$$\Phi \Leftrightarrow \Theta$$

zapíšeme vyjádřením

$$(\Phi \wedge \Theta) \vee (\neg\Phi \wedge \neg\Theta).$$

Z toho důvodu není nutné obě operace zahrnovat přímo do definice 2.2.1. ♦

Poznámka: S logickými operacemi konjunkce a disjunkce jsme se v textu zatím setkali jen v případech, kdy spojovaly dvě formule, obecněji lze ale obě logické operace zapsat též ve tvarech s libovolným počtem formulí.

Tudíž zápis ve tvaru

$$\bigwedge_{i=1}^{n} \Phi_i$$

budeme chápat jako konjunkci n formulí Φ_i, $1 \leq i \leq n$, přičemž jiným způsobem ji lze psát také jako

$$\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_i \wedge \ldots \wedge \Phi_n,$$

a zápis

$$\bigvee_{i=1}^{n} \Phi_i$$

chápeme jako disjunkci n formulí Φ_i, $1 \leq i \leq n$, s ekvivalentním tvarem

$$\Phi_1 \vee \Phi_2 \vee \ldots \vee \Phi_i \vee \ldots \vee \Phi_n. \blacklozenge$$

Přesné znění vět jakož i jejich důkazy a bližší informace k problematice nalezne čtenář například v [TAR57].

---

[4] V prvním příkladě nemůže být proměnná x nazvána volnou proměnnou, protože se v zápisu nevyskytuje, ve druhém případě je pak svázána s obecným kvantifikátorem.

3 Historický rámec a využití výpočetní techniky

Mluvíme-li o tématu eliminace kvantifikátorů a obecně o jakékoliv odbornější matematické problematice a o využívání výpočetní techniky k jejímu řešení, a to nejen ve školním prostředí, je nutné si uvědomit důležitost vztahu mezi oběma obory – matematikou a výpočetní technikou. Oba obory mají určitým způsobem symbiotický vztah a to ve větší míře, než se vyskytuje u jiných vědních oblastí. Tento vztah je charakteristický nejen vzájemným využíváním poznatků, možností a schopností jednoho oboru v oboru druhém, ale také vzájemnou potřebností.

Výpočetní technika od svého vzniku a vyčlenění z ostatních oborů staví na matematických znalostech, které umožňují algoritmizaci a výpočty, spoluvytváří samotný základ, na němž a díky němuž výpočetní stroje dokáží pracovat.

Na druhé straně v matematice nalezneme mnohé problémy a úlohy, které jsou lidským přístupem těžko anebo zdlouhavě řešitelné (patří mezi ně právě eliminace kvantifikátorů, nebo například sumace řad, řešení diferenciálních rovnic a práce s nimi, atd.), a v tomto bodě přichází na řadu matematické výpočetní programy, které takovouto práci ulehčí a urychlí. V zásadě je tedy vhodné, a nelze se tomu ani ubránit, seznamovat žáky a studenty nejen s matematickými prostředky výpočetní techniky již od počátků studia. Ty lze, jak uvidíme v kapitole 3.1, v podstatě rozdělit na dvě větší skupiny. Jednu část tvoří tzv. výuková, či učící prostředí, kdy je žák počítačovým programem vzděláván, program mu poskytuje zpětnou vazbu, zkouší jej a kontroluje žákovy dosažené výsledky, druhá část pak sestává z počítačových programů, které tvoří doplněk tradičního vzdělávacího procesu, umožňují zjednodušit práci a řešení zadaných úkolů v daném oboru. V dalších kapitolách se budeme věnovat právě této části softwaru, konkrétněji programům počítačové algebry, mezi které patří již zmíněné programy Wolfram Mathematica, Maple a QEPCAD.

Podívejme se ve stručnosti, jak zavádění a využívání prostředků výpočetní techniky do škol probíhalo v minulosti.

3.1 Výpočetní technika ve vzdělávání

První vyučovací stroj se objevil již ve dvacátých letech minulého století, v roce 1924 jej sestrojil americký psycholog Sidney L. Pressey z Ohijské státní univerzity. Šlo v podstatě o jednoduchý testovací stroj nabízející k zadaným úlohám výběr z několika možných odpovědí. Zajímavá je pohnutka, která jej k sestavení stroje vedla – původně

si chtěl ulehčit práci s opakovaným zkoušením velkého množství studentů, teprve později zjistil, že studenti se používáním stroje také učí. Tento počin dal tedy vzniknout Presseyho lineárnímu programu (s výběrovou odpovědí), jak je dnes tento systém nazýván. Podstatnou nevýhodou však bylo, že stroj podával testovanému subjektu pouze jednoduchou zpětnou vazbu ve smyslu „správně“ – „špatně“, o kvalitě chyby se už student nedozvěděl nic.

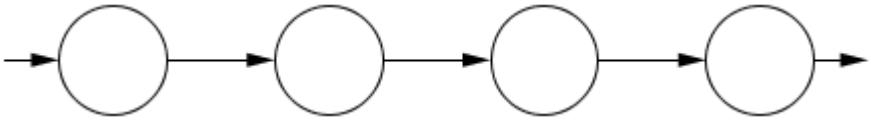
Následovalo zdokonalování takovýchto jednoduchých strojů, přičemž některé byly určeny i pro testování komplexnějších problémů, přičemž stroje dokázaly hodnotit správnost jednotlivých kroků celkového řešení.

V padesátých letech pak docházelo, spolu s rozvojem číslicových počítačů a automatizace, k vytvoření tzv. programového učení. Jeho otcem je další americký psycholog Burrhus F. Skinner, který prosazuje neobehavioralistickou teorii učení. Podle ní se člověk (ale obecně jakýkoliv organismus) učí lépe, pokud je jeho chování formováno sledem dílčích učebních aktivit a postupně tak přibližováno k cíli. Tyto dílčí aktivity přitom musí být vždy bezprostředně „zpevněny“ (jinými slovy musí být potvrzena správnost právě proběhnutého aktu), čímž se zvyšuje pravděpodobnost, že později se bude ve stejné či podobné situaci opakovat toto chování vedoucí k požadovanému cíli.

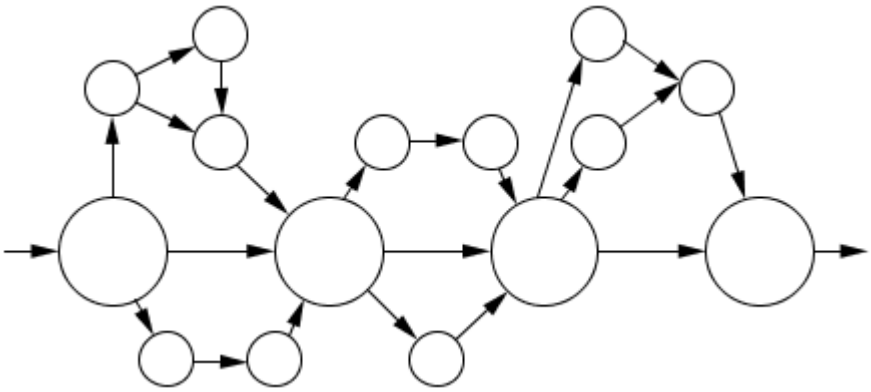
V této době byly stanoveny i hlavní pilíře programového učení, kterými jsou:

- krátké kroky – veškeré učivo je rozděleno na krátké úseky, které jsou žákům předkládány víceméně izolovaně (zřejmou nevýhodou je ztráta souvislostí a náhledu na celý problém),
- aktivní odpovědi – po žákovi je vždy vyžadována odpověď na předložený problém,
- bezprostřední „zpevnění“ – žákova odpověď je okamžitě vyhodnocena a on je informován o její správnosti,
- vlastní tempo – každý žák si může zvolit vlastní tempo studia,
- učení podle přesně vymezeného programu – všechno učení probíhá podle předem přesně stanoveného sledu kroků (nevýhodou je, že systém nepřipouští originální přístup k předloženým problémům),
- optimalizace programu – na základě používání vyučovacího programu je neustále ověřována a zlepšována jeho účinnost a kvalita.

Kromě lineárních vyučovacích programů (Presseyho s výběrovou odpovědí a Skinnerův s tvořenou odpovědí), které sestávaly z pevně dané posloupnosti dílčích kroků, které musel student projít všechny (Obrázek 6a), se objevovaly i větvené programy (například Crowderův, Obrázek 6b), které již dokázaly relativně dobře reagovat na kvalitu chyby a podle toho odkázat studenta do některé z vedlejších větví programu, kde se nacházely jednodušší úkoly. Ty měly za úkol studenta doučit, odstranit chybu, které se dopustil, a její příčiny a vrátit jej zpět do hlavní větve programu.



Obrázek 1a: Lineární program



Obrázek 1b: Větvený program

Vznik programového učení podnítil snahu využívat vyučovací stroje přímo ve výuce. Nejdále v tomto směru byly Spojené státy, kde se v šedesátých letech začaly ve školách používat vyučovací stroje v kombinaci například se zvukovými záznamy, projekcí diapozitivů ap., přičemž odpovědi testovaných žáků byly zaznamenávány prostřednictvím klávesnic. V sedmdesátých letech pak již nastoupila (skupinová) zpětnovazební zařízení, která byla sestrojena pro učení a testování i celých tříd.

Na konci sedmdesátých let došlo k velkému propadu důvěry v tuto cestu vyučování, jelikož vyučovací stroje nedokázaly splnit přehnaná očekávání, navíc jejich vývoj a sestrojování byly poměrně nákladné. Štafetu vyučovacích strojů převzala moderní výpočetní technika spolu s rozvojem umělé inteligence a kybernetiky. [MAS04, WHI06]

3.1.1 Počítačové programy jako výukové a učící prostředí

Využívání výpočetní techniky jako učebního prostředí lze rozdělit do tří etap, přičemž etapa první, v níž se využívaly velké sálové počítače se sítí individuálních terminálů, se

objevuje již v šedesátých letech 20. století. Například od roku 1962 byl v těchto systémech rozvíjen projekt PLATO (Programmed Logic for Automated Teaching Operations) používající k učení řízený dialog mezi studentem a počítačem, jakož i různé formy her. Na jeho vývoji spolupracovalo i mnoho učitelů ze Spojených států a Kanady, kteří pro něj vytvořili výukové lekce pro asi 4 tisíce hodin studia. Projekt byl poměrně úspěšný, navázaly na něj projekty PLATO II, III a IV.

V sedmdesátých letech přichází velký rozvoj minipočítačů a mikropočítačů, které nabízí nové možnosti. Zavádění nové techniky, která je finančně dostupnější než drahé sálové počítače, umožňuje skupinové nebo individuální řízení výukové činnosti odděleně od centrálního počítače, navíc se objevuje možnost zapojení nových didaktických prostředků řízených počítačem jako například videosystémů nebo různých simulátorů.

S koncem osmdesátých let přichází třetí etapa vyznačující se užíváním nejprve lokálních a posléze globálních počítačových sítí. Tato propojenost má za následek sjednocování do té doby velmi různorodých systémů, začínají se objevovat první multimediální encyklopedie i online kurzy, ulehčujících individuální studium a distanční formu vzdělávání. S tímto rozvojem pak úzce souvisí masové využívání audiovizuální komunikace, uplatnění nejrůznějších sociálních sítí (Facebook, Twitter) a rozvoj systémů virtuální reality (Second Life), jejichž rozšíření do škol lze předpokládat v následujících letech. [MAS04, WHI06]

Používání počítačů a softwarového vybavení jako učebního prostředí však není jediným způsobem jejich zakomponování do vzdělávacího procesu. Více důležité je učení práci s jednotlivými programy a využívání jich k zjednodušení řešených úkolů a problémů. V našem případě nás budou zajímat programy matematické, konkrétně sofistikované programy počítačové algebry.

3.1.2 Počítačové programy jako prostředek

Neméně důležitý způsob využívání výpočetní techniky jsou případy práce s počítačovým programem jako silným pomocníkem ulehčujícím složité výpočty, řešení matematických problémů či vyšetřování parametrických systémů, konkrétně tedy programy počítačové algebry a výpočetní prostředí.

Mezi nejznámější aplikace z této skupiny patří bezesporu již zmíněná Wolfram Mathematica, dále pak program Maple od společnosti Maplesoft, či Matlab od firmy MathWorks.

Mluvíme-li o těchto složitých programech, které díky svým výpočetním možnostem a schopnostem usnadňují práci lidem napříč nejrůznějšími zaměstnáními, počínaje matematiky, statistiky, pokračujíc přes inženýry a konče například právě učiteli matematiky a jejich studenty, je nutné si uvědomit, na jakém základě tyto programy pracují. Jde o zajímavé a přínosné spojení matematiky a výpočetní techniky, bez něhož by tato (a i mnohá podobná) problematika byla neřešitelná.

V zásadě jde o to, že matematika přináší svůj matematický aparát, nabízí své výpočetní postupy a umožňuje provádět často velice složité výpočty, které jsou však dosti často bez účinné pomoci takřka nepoužitelné.

Tuto účinnou pomoc v tomto případě poskytuje právě výpočetní technika v podobě výkonných počítačů a algoritmizovanými matematickými postupy.

Z tohoto symbiotického vztahu matematiky a výpočetní techniky pak vyplývá zajímavý a užitečný řešitelský přístup, kterého by každý z obou oborů o samotě nebyl schopen.

3.2 Alfred Tarski

Jak již bylo napsáno v úvodu, patrně nejvýznamnější osobností v oblasti eliminace kvantifikátorů v matematických formulích a jejího využití k řešení problémů byl Alfred Tarski, vynikající matematik a logik, který dokázal, že eliminace kvantifikátorů v reálně uzavřených tělesech je možná.

Obrázek 2a: Alfred Tarski (1902 – 1983)

Alfred Tarski (původním jménem Alfred Tajtelbaum, někdy též Taitelbaum) se narodil 14. ledna 1902 do rodiny židovského obchodníka Ignáce Tajtelbauma a jeho ženy Rosy. Oblast Mazovského vojvodství s hlavním městem Varšavou, kde rodina žila, patřila od roku 1772 (Trojí dělení Polska) pod vliv imperiálního Ruska, jehož vláda uplatňovala proti polským občanům všemožné represe. Z tohoto důvodu byla zrušena Varšavská

univerzita a její místo zaujala univerzita ruskojazyčná. V polovině roku 1915 se však ruská armáda musela z Polska stáhnout před postupujícími vojsky Německa a Rakouska-Uherska a polská univerzita mohla být znovu otevřena. S přispěním polských matematických kapacit Jana Łukasiewicze[5], Stefana Mazurkiewicze[6] a dalších se brzy stala věhlasnou institucí na poli matematiky. V té době Alfred dokončil svá středoškolská studia a po krátkém působení v polské armádě nastoupil v roce 1918 ke studiu biologie. Na přednáškách z logiky si jej ale všiml další z učitelů Stanisław Leśniewski[7] a podařilo se mu ho přesvědčit, aby přestoupil z biologie na studium matematiky, kde se začal zabývat otázkami týkajícími se teorie množin, kteréžto problematice se pak kromě jiného věnoval po zbytek života.

Následně nastoupil do doktorského studia, v jehož průběhu mu byl školitelem právě Leśniewski. V roce 1924 Tarski promoval a ve svých 22 letech se stal nejmladším absolventem tohoto stupně vysokoškolského vzdělání Varšavské univerzity. Již před ukončením doktorského studia Alfred přemýšlel o své budoucí akademické kariéře a bylo mu zřejmě jasné, že v tehdejší Evropě se sílícími tendencemi k antisemitismu jako žid nebude v lehkém postavení při získávání dobrého místa na univerzitě. To byl také, spolu se silným patriotistickým cítěním, důvod, proč si se svým bratrem Waclawem nechal změnit příjmení z původního Tajtelbaum na více polsky znějící a též lehčeji vyslovitelné Tarski. Zároveň s touto změnou též konvertoval k římskokatolickému křesťanství.

Z tohoto období pochází Tarského první větší díla a výsledky. Zabýval se například rozvojem výsledků Cantora, Zermela a Dedekinda v oblasti teorie množin a se Stefanem Banachem publikovali společný článek o důkazu tzv. Banach-Tarského paradoxu, který říká, že koule může být rozdělena na konečný počet částí a pak znovu sestavena do

---

[5] Jan Łukasiewicz (1878-1956) byl významný polský logik a filozof, profesor univerzit ve Lvově a ve Varšavě. Mezi světovými válkami zastával post polského ministra školství a též děkana a později rektora Varšavské univerzity, po druhé světové válce se odstěhoval do Dublinu, kde se stal členem Královské irské akademie.

[6] Stefan Mazurkiewicz (1888-1945) byl polským matematikem zabývajícím se oblastmi matematické analýzy, topologie a pravděpodobnosti, působil jako profesor na univerzitě ve Varšavě, byl členem Polské akademie věd. Zajímavá kapitola jeho života spadá do období polsko-sovětské války (1919-1921), kdy se Mazurkiewicz podílel na rozluštění ruského vojenského šifrování.

[7] Stanisław Leśniewski (1886-1939) byl polský logik, filozof a matematik vyučující na Varšavské univerzitě, pozdější vedoucí doktorských studií Alfreda Tarského.

koule o větší velikosti (případně do dvou koulí, jejichž velikost je shodná s velikostí koule původní). Jde o paradoxní důsledek axiomu výběru.

Mezi léty 1922 a 1925 vyučoval Tarski logiku na Polském pedagogickém institutu ve Varšavě, byl jmenován docentem na Varšavské univerzitě a později se stal Łukasiewiczovým asistentem. I přes působení na těchto významných postech se mu ale nedostávalo dostatečného množství peněz a tak musel přijmout jako druhé zaměstnání místo učitele matematiky na Zeromského lyceu, kde se seznámil se svou budoucí manželkou Marií Witkowskou, s níž měl dvě děti – Jana a Inu.

V následujících letech se nadále věnoval své práci v oblasti matematiky a logiky, navštívil pracovně Vídeň a Paříž a spolupracoval s matematiky sdruženými okolo Karla Mengera. Ještě před vypuknutím druhé světové války se v roce 1939 pokoušel získat posty na univerzitách ve Lvově a v Poznani, ale ani v jednom případě neuspěl. Přestože byl v té době Tarski ve světě velmi uznávaný, svou úlohu zde pravděpodobně sehrál i jeho židovský původ, jelikož antisemitské nálady v polské společnosti té doby byly velmi výrazné.

I přes tuto nepřízeň osudu měl Tarski nakonec štěstí. V srpnu 1939 odjel do Spojených států na konferenci pořádanou na Harvardské univerzitě, což mu nepochybně zachránilo život, a přečkal zde celou válku. Bohužel podcenil nebezpečí ze strany nacistického Německa a zanechal v Polsku celou svou rodinu, a přestože se s pomocí mnoha svých přátel snažil zorganizovat převoz své manželky i dětí z okupované vlasti do USA, neuspěl a setkal se s nimi až po skončení války v roce 1946. Nicméně většina jeho rodiny, včetně otce, matky, bratra Waclawa a švagrové, za války zemřeli.

I když si Tarski odjezdem z Evropy zachránil život, neměl ve Spojených státech snadnou pozici, jelikož spolu s ním přišlo i mnoho dalších akademických pracovníků, kteří utekli před nacismem a válkou. Musel se tak spokojit s řadou dočasných postů, které zastával: v letech 1939 až 1941 pracoval v Harvardu, v roce 1940 přijal místo na City College v New Yorku a od roku 1941 do roku 1942 působil v Princetonu na Institute for Advanced Study. Teprve až roku 1942 dostal nabídku pracovat na University of California v Berkeley, kde dosáhl postu profesora matematiky a strávil zde zbytek své kariéry. V roce 1973 měl odejít do důchodu, avšak přijal žádost, aby na škole zůstal. Během těchto let strávených ve Spojených státech na univerzitě v Berkeley pokračoval Tarski ve své práci, dělal školitele 24 studentům doktorského studia, mezi nimiž bylo na svou dobu poměrně vysoké procento žen, a využil mnoha příležitostí k zahraničním cestám, přičemž přednášel například na University College v Londýně,

Institutu Henriho Poincaré v Paříži nebo Pontificia Universidad Católica v Chile. Za své zásluhy na poli vědy byl vyznamenán mnoha čestnými doktoráty a stal se členem Národní akademie věd Spojených států amerických či British Academy (státní akademie Spojeného království pro humanitní a společenské vědy).
Alfred Tarski zemřel v Berkeley 26. října 1983. [FEF06, OCR03, WTC09, WTE09]

3.3 Tarského práce

V době, kdy se Tarski problematice eliminace kvantifikátorů začal věnovat, však nemohl ke svému zkoumání využít prostředky výpočetní techniky a to ze dvou hlavních důvodů. Za prvé dosud neexistovaly počítače, jak je známe dnes, jejich vývoj a především zapojování do běžného života započal až později a probíhal z počátku relativně pomalu, přičemž první objevivší se stroje neměly dostatečný výkon pro provádění potřebných složitých matematických operací. Druhý důvod pak spočíval v neexistenci příslušného využitelného algoritmu.
Přesněji řečeno jisté algebraické algoritmy v tomto směru již byly dostupné, sám Tarski ten svůj založil na zobecněné rozšířené Sturmově větě, pomocí níž je možné zjistit počet kořenů daného polynomu v určeném intervalu. Avšak tento přístup byl vcelku složitý a jeho náročnost nesnížila ani vylepšení aplikovaná Abrahamem Seidenbergem a později Paulem Cohenem.
Tarski tak v této době ani nepředpokládal, že by celý proces eliminace mohl být algoritmizovatelný do té míry, aby byl prakticky využitelný.
Eliminace kvantifikátorů tak byla za těchto podmínek často neřešitelným nebo přinejmenším nadmíru komplikovaným problémem, což Tarski vnímal jako nepřekonatelnou překážku pro efektivní praktické využití celého přístupu a proto své snahy a výzkumy v této oblasti značně omezil.
I přesto svou práci na procesu eliminace kvantifikátorů považoval za jeden z nejdůležitějších podniků svého života. Na druhou stranu až překvapivě dlouho nečinil žádné přípravy k vydání svých objevů. Teprve až v roce 1939 začal s přípravami vydání článku *The completeness of elementary algebra and geometry* v pařížském vydavatelství, avšak nakonec kvůli německé invazi do Francie na jaře a v létě roku 1940 k publikování vůbec nedošlo. Tarskému v té době z celého článku zůstalo jen několik málo stránek s důkazy.
S další prací na přípravách vydání započal Tarski až v roce 1948. Jeho přítel a kolega J. C. C. McKinsey, tehdy pracující ve společnosti RAND v Santa Monice, přesvědčil své

nadřízené o potenciálu a možném přínosu aplikování Tarského poznatků v počítačových výpočtech (hlavní důraz byl kladen na využitelnost v optimalizaci strategií v některých hrách, přičemž teorie her byla v té době poměrně populární).

To však znamenalo rozpracovat detailně veškerou potřebnou teorii, a tak McKinsey pod Tarského vedením zrevidoval veškeré zbývající podklady z roku 1939 a v roce 1948 se práce objevila pod novým názvem *A decision procedure for elementary algebra and geometry*. O tři roky později pak byla vydána veřejně nakladatelstvím UC Press pod stejným názvem.

Zajímavou skutečností z pohledu dnešního čtenáře je fakt, že Tarski v této své práci zvažuje využití jakéhosi automatického stroje, který by na základě příslušného předpisu (tedy algoritmu, i když toto pojmenování v textu není nikde použito) dokázal eliminaci kvantifikátorů provádět bez zásahu uživatele.

Dodejme ještě, že změna v názvu obou prací v letech 1939 a 1948/1951 souvisí se změnou cílů, které si Tarski a jeho spolupracovníci kladli. [FEF06]

3.4 Tarského eliminační postup

Projděme stručně zmíněný postup tak, jak je popsán v Tarského knize *A Decision Method for Elementary Algebra and Geometry* vzešlé z Projektu RAND.

Celý postup eliminace kvantifikátorů sestává ze dvou samostatných částí, v první části je možné se mechanickým způsobem dopracovat od kvantifikované formule Θ k ekvivalentní formuli bez kvantifikátorů, přičemž tato nová formule neobsahuje jiné volné proměnné než ty, které se vyskytovaly ve formuli Θ, druhá část postupu pak dovoluje rozhodnout, zda daná formule bez kvantifikátorů je pravdivá, či ne.

V textu se neobejdeme bez několika definic, jejichž prostřednictvím jsou zavedeny potřebné operátory.

Definice 3.4.1: Nechť polynom

$$p = p_n \cdot x^n + \ldots + p_1 \cdot x + p_0$$

je polynom v proměnné x stupně n. Potom prvním reduktem budeme rozumět polynom získaný z polynomu p vynecháním členu $p_n \cdot x^n$ a zapisujeme jej

$$Rd_x(p) = p_{n-1} \cdot x^{n-1} + \ldots + p_1 \cdot x + p_0.$$

Je-li $n = 0$, platí rovnost

$$Rd_x(p) = 0.$$

Obecně pak redukta libovolného řádu definujeme rekurentně:

$$Rd_x^{\,0}(p) = p,$$

$$Rd_x^{\,k+1}(p) = Rd_x[Rd_x^{\,k}(p)].\blacklozenge$$

Definice 3.4.2: Dále definujme $P_x(p)$, pro který bude platit:

a) Je-li $p = x$, pak je $P_x(p) = 1 \cdot x$,

b) je-li p konstanta z množiny $\{-1; 0; 1\}$ nebo proměnná různá od x, pak je $P_x(p) = p$,

c) jsou-li p a q dva libovolné algebraické členy, pak je $P_x(p + q) = P_x(p) + P_x(q)$ a $P_x(p \cdot q) = P_x(p) \cdot P_x(q).\blacklozenge$

Takto definovaný předpis P_x pro libovolné algebraické členy je polynomem. Je též vhodné jej definovat i pro všechny matematické formule bez kvantifikátorů.

Definice 3.4.3: Je-li Θ matematická formule bez kvantifikátorů a x proměnná, potom $P_x(\Theta)$ je ekvivalentní s Θ, přičemž $P_x(\Theta)$ je matematická formule tvořená atomickými formulemi ve tvarech $p > 0$ a $p = 0$, kde p je polynom v proměnné x, pomocí logických spojek $\wedge$ a $\vee$ (nikoliv však $\neg$).$\blacklozenge$

Definice 3.4.4: Pro všechny algebraické členy p a q a všechny matematické formule Θ a Φ platí:

a) $P_x(p = q)$ je ekvivalentní s $P_x(p - q) = 0$,

b) $P_x(p > q)$ je ekvivalentní s $P_x(p - q) > 0$,

c) $P_x[\neg(p = q)]$ je ekvivalentní s $[P_x(p > q) \vee P_x(p < q)]$,

d) $P_x[\neg(p > q)]$ je ekvivalentní s $[P_x(p = q) \vee P_x(p < q)]$,

e) $P_x(\Theta \vee \Phi)$ je ekvivalentní s $[P_x(\Theta) \vee P_x(\Phi)]$,

f) $P_x(\Theta \wedge \Phi)$ je ekvivalentní s $[P_x(\Theta) \wedge P_x(\Phi)]$,

g) $P_x[\neg(\Theta \vee \Phi)]$ je ekvivalentní s $P_x(\neg\Theta) \vee P_x(\neg\Phi)$,

h) $P_x[\neg(\Theta \wedge \Phi)]$ je ekvivalentní s $P_x(\neg\Theta) \wedge P_x(\neg\Phi)$,

i) $P_x(\neg\neg\Theta)$ je ekvivalentní s $P_x(\Theta).\blacklozenge$

Užitím výše uvedeného dostaneme pro každou formuli Θ bez kvantifikátorů ekvivalentní formuli $\Psi \equiv P_x(\Theta)$, která je také bez kvantifikátorů a zároveň neobsahuje negace.

Definujme též operátor Q, pomocí nějž transformujeme libovolnou formuli Ψ tohoto typu do tzv. disjunktního normálního tvaru.

Definice 3.4.5: Je-li Φ atomická formule, potom

$$Q(\Phi) \equiv \Phi.$$

Jsou-li

$$Q(\Phi_1) \equiv \bigvee_{i \le m} \bigwedge_{j \le m_i} \Psi_{i,j}$$

a

$$Q(\Phi_2) \equiv \bigvee_{m < i \le m+n} \bigwedge_{j \le m_i} \Psi_{i,j},$$

kde $\Psi_{i,j}$, $i \le m + n$ a $j \le m_i$, je atomická formule, pak

$$Q(\Phi_1 \vee \Phi_2) \equiv \bigvee_{i \le m+n} \bigwedge_{j \le m_i} \Psi_{i,j}$$

a

$$Q(\Phi_1 \wedge \Phi_2) \equiv \bigvee_{\substack{i \le m \\ m < j \le m+n}} \left(\psi_{i,1} \wedge \ldots \wedge \psi_{i,m_i} \wedge \psi_{j,1} \wedge \ldots \wedge \psi_{j,m_j}\right). \blacklozenge$$

Platí, že pokud formule Φ neobsahuje kvantifikátory ani negace, potom je $Q(\Phi)$ zapsána ve tvaru disjunkce konjunkcí atomických formulí a je s formulí Φ ekvivalentní.

Spolu s předchozím budeme potřebovat též derivaci polynomu, kterou připomeneme v definici. V tomto případě se nejedná o derivaci definovanou prostředky matematické analýzy, ale o tzv. formální derivaci polynomu užívanou v algebře.

Definice 3.4.6: Mějme polynom $p = p_n \cdot x^n + ... + p_1 \cdot x + p_0$ stupně n v proměnné x, potom předpis $D_x(p) = n \cdot p_n \cdot x^{n-1} + ... + 2 \cdot p_2 \cdot x + p_1$ nazýváme derivace polynomu p. Pokud je polynom p v proměnné x stupně 0, pak derivace $D_x(p) = 0$.

Derivace libovolných vyšších řádů mohou být zavedeny rekurentně:

$$D_x^0(p) = p,$$
$$D_x^{k+1}(p) = D_x[D_x^k(p)]. \blacklozenge$$

Představme operátor M, pro který existuje formule $M_x^n(p)$, kde p je libovolným polynomem stupně n v proměnné x a kterou budeme chápat jako vyjádření ve smyslu, že x je kořenem polynomu p. V případě, že $n = 0$, tato formule znamená, že x není kořenem p. Bez ohledu na to, zda platí $n = 0$ nebo $n > 0$, formuli $M_x^n(p)$ čteme „číslo x je kořen násobnosti n v polynomu p“.

Definice 3.4.7: Nechť p je polynom v proměnné x. Pokud je $n > 0$, pak $M_x^n(p)$ je ekvivalentní se zápisem

$$\{[\bigwedge_{1 \le i \le n} (D_x^{i-1}(p) = 0)] \wedge \neg[(D_x^n(p) = 0)]\}$$

a dále

$$M_x^{\,0}(p) \equiv \neg(p = 0).\blacklozenge$$

Využijeme také typ existenčního kvantifikátoru $\exists_n$, který jsme představili v kapitole 2.2. Připomeňme jen, že zápis $(\exists_n x)$: Φ znamená, že existuje právě n hodnot proměnné x, pro něž je formule Φ pravdivá.

Pokud je n kladné celé číslo a n proměnných $\eta_1, ..., \eta_n$ z posloupnosti všech proměnných je různých od x a nevyskytuje se ve formuli Φ, platí ekvivalentní zápis

$$(\exists_n x)\colon \Phi \equiv \{(\exists \eta_1)...(\exists \eta_n)\colon (\bigwedge_{1\le i<j\le n} \neg(\eta_i = \eta_j) \wedge (\forall x)\colon [\Phi \Leftrightarrow \bigvee_{1\le i\le n} (\eta_i = x)])\}.$$

Takto zavedený kvantifikátor využijeme při představení operátoru F. Je-li n celé číslo, x proměnná a p a q libovolné polynomy, pak $F_x^{\,n}(p, q)$ je formule chápaná ve smyslu, že existuje právě n hodnot x, které splňují podmínky:

a) x je kořenem vyššího stupně polynomu p než polynomu q, přičemž rozdíl mezi těmito stupni je liché celé číslo,

b) existuje otevřený interval, jehož pravou mezní hodnotou je x a polynomy p a q v něm mají stejná znaménka.

Definice 3.4.8: Mějme polynom p v proměnné x stupně r a polynom q v proměnné x stupně s. Pokud jsou η_1 a η_2 dvě první proměnné s posloupnosti proměnných různých od x, které se nevyskytují v polynomech p a q, pak platí

$$F_x^{\,n}(p, q) \equiv (\exists_n x)\colon \{\bigvee_{\substack{0\le k\le s\\ 0\le 2m\le r-k-1}} [M_x^{\,k+2m+1}(p) \wedge M_x^{\,k}(q)] \wedge$$

$$(\exists \eta_1)(\exists \eta_2)\colon [(\eta_1 = x) \wedge (x > \eta_2) \wedge (\forall x)\colon \{[(x > \eta_2) \wedge (\eta_1 > x)] \Rightarrow (p\cdot q > 0)\}]\}.\blacklozenge$$

S právě definovaným operátorem F úzce souvisí operátor G, který pro dva polynomy p a q v proměnné x píšeme $G_x^{\,n}(p, q)$.

Definice 3.4.9: Nechť n je libovolné celé číslo (kladné, záporné nebo nulové), p a q jsou polynomy v proměnné x a k je maximum stupně polynomů p a q, pak

$$G_x^{\,n}(p, q) \equiv \bigvee_{\substack{0\le m\le k\\ 0\le m+n\le k}} [F_x^{\,n+m}(p, q) \wedge F_x^{\,m}(p, -q)].\blacklozenge$$

Jinými slovy n je takové celé číslo, pro které platí $n = n_1 - n_2$, kde n_1 celé číslo, pro které platí $F_x^{\,n_1}(p,q)$, a n_2 je také celé číslo, pro něž platí $F_x^{\,n_2}(p,-q)$.

Budeme také potřebovat zavedení zbytku po dělení polynomu polynomem, resp. pro naše potřeby bude vhodnější pracovat s polynomem opačným ke zbytku.

Definice 3.4.10: Mějme polynom p stupně m a s vedoucím koeficientem p_m a polynom q stupně n s vedoucím koeficientem q_n, oba v proměnné x. Polynom opačný ke zbytku po jejich dělení potom označíme $Z_x(p, q)$ a platí pro něj:

a) $Z_x(p, q) = -p$, pokud $m < n$,

b) $Z_x(p, q) = Rd_xP_x(p_m \cdot q_n \cdot q - p \cdot q_n^2)$, pokud $m = n$,

c) $Z_x(p, q) = Z_x[Rd_xP_x(p \cdot q_n^2 - p_m \cdot q_n \cdot x^{m-n} \cdot q), q]$, pokud $m > n$. ♦

Je zřejmé, že výsledný polynom $Z_x(p, q)$ dvou polynomů p a q bude také polynomem, přičemž se v něm budou vyskytovat pouze proměnné, které se vyskytují i v p a q. Dále pro pořádek připomeňme, že když je polynom q stupně $n > 0$ v proměnné x, pak stupeň výsledného polynomu $Z_x(p, q)$ je menší než n, a je-li stupeň polynomu q roven $n = 0$ v proměnné x, potom $Z_x(p, q) = 0$.

Poznámka: Tradičně je v algebře zbytek po dělení polynomu polynomem definován trochu odlišně, a sice pomocí dělení polynomu p polynomem q, kdy dostaneme rovnost

$$p = a \cdot q + z,$$

kde polynom a představuje podíl p/q a polynom z je právě hledaný zbytek. Oba tyto polynomy, a a z, je možné spočítat pomocí operací sčítání, odčítání, násobení a dělení. Výše uvedená definice se od tohoto postupu kvůli možným komplikacím s operací dělení liší. Z tohoto důvodu nelze jednoduše určit polynom a. ♦

V dalším textu, který je těžištěm prezentované myšlenky, představíme tři operátory S, T, U, které uvádí do vztahu formule $S(\Phi)$, $T(\Phi)$ a $U(\Phi)$ bez kvantifikátorů pro formuli Φ. Pro tyto nové formule platí, že jsou ekvivalentní s původní formulí Φ.

Operátor S je definován pro speciální formule ve tvaru $G_x^k(p, q)$, operátor T, konstruovaný pomocí operátoru S, je definován pro širší třídu složitějších formulí, například ve tvarech $(\exists_k x)$: $(p = 0)$, $(\exists_k x)$: $[(p > 0) \vee (q = 0)]$, a operátor U zkonstruovaný podle operátoru T je definovaný pro všechny formule.

Přestože všechny tři operátory S, T a U spolu souvisí (U je konstruován pomocí T a tedy nepřímo pomocí S) a formule $S(\Phi)$, $T(\Phi)$ a $U(\Phi)$ jsou ekvivalentní, nejsou tyto formule totožné.

Definice 3.4.11: Nechť k je celé číslo, p a q polynomy v proměnné x stupňů m a n s vedoucími koeficienty p_m a q_n a $\Phi \equiv G_x^k(p, q)$. Potom platí

$$S(\Phi) \equiv (0 = 0) \text{ pro } k = 0$$

a

$$S(\Phi) \equiv (0 = 1) \text{ pro } k \neq 0,$$

je-li některý z polynomů p, q roven nule,

$$S(\Phi) \equiv \{[(p_m = 0) \wedge SG_x^{\,k}(Rd_x(p), q)] \vee [(q_n = 0) \wedge SG_x^{\,k}(p, Rd_x(q))] \vee$$
$$[\neg(p_m \cdot q_n = 0) \wedge SG_x^{\,k}(q, R_x(p, q))]\},$$

není-li ani jeden z polynomů p, q roven nule a $m + n$ je sudé, a konečně

$$S(\Phi) \equiv \{[(p_m = 0) \wedge SG_x^{\,k}(Rd_x(p), q)] \vee [(q_n = 0) \wedge SG_x^{\,k}(p, Rd_x(q))] \vee$$
$$[(p_m \cdot q_n > 0) \wedge SG_x^{\,k+1}(q, R_x(p, q))] \vee$$
$$[(p_m \cdot q_n < 0) \wedge SG_x^{\,k-1}(q, R_x(p, q))]\},$$

není-li ani jeden z polynomů p, q roven nule a $m + n$ je liché. ♦

Definice 3.4.12: Pro libovolné polynomy ve tvarech

$$p = p_m \cdot x^m + \ldots + p_1 \cdot x + p_0,$$
$$q = q_n \cdot x^n + \ldots + q_1 \cdot x + q_0,$$
$$\gamma_1 = \gamma_{1,n_1} \cdot x^{n_1} + \ldots + \gamma_{1,1} \cdot x + \gamma_{1,0},$$
$$\ldots$$
$$\gamma_r = \gamma_{r,n_r} \cdot x^{n_r} + \ldots + \gamma_{r,1} \cdot x + \gamma_{r,0}$$

v proměnné x definujeme operátor T následovně:

a) pro formuli Φ ve tvaru

$$(\exists_k x)\colon (p = 0)$$

je

$$T(\Phi) \equiv [\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge SG_x^{\,-k}(p, D_x(p)),$$

b) pro formuli Φ ve tvaru

$$[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge (\exists_k x)\colon [(p = 0) \wedge (q > 0)]$$

je

$$T(\Phi) \equiv \{[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge \bigvee_{\substack{2k = r_1 - r_2 + r_3 \\ 0 \le r_1, r_2 \le m \\ -m \le r_3 \le m}} [\, SG_x^{\,-r_1}(p, D_x(p)) \wedge$$
$$SG_x^{\,-r_2}(P_x(p^2 + q^2), D_x P_x(p^2 + q^2)) \wedge$$
$$SG_x^{\,-r_3}(p, D_x(p) \cdot q)]\},$$

c) pro formuli Φ ve tvaru

$$[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge (\exists_k x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge \ldots \wedge (\gamma_r > 0)],$$

kde $r \geq 2$, je

$$T(\Phi) \equiv \bigvee_{\substack{2k=r_1+r_2-r_3 \\ 0\le r_1,r_2,r_3\le m}} [\, T(\Phi_1) \wedge T(\Phi_2) \wedge T(\Phi_3)],$$

přičemž

$$\Phi_1 \equiv \{[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge$$
$$(\exists_{r_1} x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge \ldots \wedge (\gamma_{r-2} > 0) \wedge$$
$$P_x(\gamma_{r-1}\cdot\gamma_r^2) > 0]\},$$
$$\Phi_2 \equiv \{[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge$$
$$(\exists_{r_2} x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge \ldots \wedge (\gamma_{r-2} > 0) \wedge$$
$$P_x(\gamma_{r-1}{}^2\cdot\gamma_r) > 0]\} \text{ a}$$
$$\Phi_3 \equiv \{[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge$$
$$(\exists_{r_3} x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge \ldots \wedge (\gamma_{r-2} > 0) \wedge$$
$$P_x(-\gamma_{r-1}\cdot\gamma_r) > 0]\},$$

speciálně pro případ $r = 2$ vynecháme ve formulích Φ_1, Φ_2, Φ_3 část konjunkce

$$(\gamma_1 > 0) \wedge \ldots \wedge (\gamma_{r-2} > 0),$$

d) pro formuli Φ ve tvaru

$$\neg(p_m = 0) \wedge (\exists_k x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge \ldots \wedge (\gamma_r > 0)]$$

je

$$T(\Phi) \equiv (\neg(p_m = 0) \wedge T\{[\neg(p_m = 0) \vee \ldots \vee \neg(p_0 = 0)] \wedge$$
$$(\exists_k x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge \ldots \wedge (\gamma_r > 0)]\}),$$

e) pro formuli Φ ve tvaru

$$[\neg(\gamma_{1,n_1} = 0) \wedge \ldots \wedge \neg(\gamma_{r,n_r} = 0)] \wedge (\exists_k x)\colon [(\gamma_1 > 0) \wedge \ldots \wedge (\gamma_r > 0)]$$

je

$$T(\Phi) \equiv (0 = 1)$$

pro $k > 0$,

$$T(\Phi) \equiv \{[\neg(\gamma_{1,0} = 0) \wedge \ldots \wedge \neg(\gamma_{r,0} = 0)] \wedge [(0 > \gamma_{1,0}) \vee \ldots \vee (0 > \gamma_{r,0})]\}$$

pro $k = 0$ a $n_1 + \ldots + n_r = 0$,

$$T(\Phi) \equiv \{\neg(\gamma_{1,n_1} = 0) \wedge \ldots \wedge \neg(\gamma_{r,n_r} = 0) \wedge$$
$$[(0 > \gamma_{1,n_1}) \vee \ldots \vee (0 > \gamma_{r,n_r})]\} \wedge$$
$$\{[0 > (-1)^{n_1}\cdot\gamma_{1,n_1}] \vee \ldots \vee [0 > (-1)^{n_r}\cdot\gamma_{r,n_r}]\} \wedge$$
$$T\{\neg(\delta = 0) \wedge (\exists_0 x)\colon ([D_x P_x(\gamma_1\cdot\ldots\cdot\gamma_r) = 0] \wedge$$
$$(\gamma_1 > 0) \wedge \ldots \wedge (\gamma_r > 0))\}$$

pro $k = 0$ a $n_1 + ... + n_r > 0$, kde δ je vedoucí koeficient $D_x P_x(\gamma_1 \cdot ... \cdot \gamma_r)$,

f) pro formuli Φ ve tvaru

$$(\exists_k x)\colon [(\gamma_1 > 0) \wedge ... \wedge (\gamma_r > 0)]$$

je

$$T(\Phi) \equiv (0 = 1)$$

pro $k \neq 0$,

$$T(\Phi) \equiv \{[(\gamma_{1,0} = 0) \wedge ... \wedge (\gamma_{1,n_1} = 0)] \vee ... \vee [(\gamma_{r,0} = 0) \wedge ... \wedge (\gamma_{r,n_r} = 0)]\} \vee$$

$$\bigvee_{(s_1,...,s_r) \in S} \{\psi_{1,s_1} \wedge ... \wedge \psi_{r,s_r} \wedge T([\neg(\gamma_{1,s_1} = 0) \wedge ... \wedge \neg(\gamma_{r,s_r} = 0)] \wedge$$

$$(\exists_0 x)\colon [(Rd_x^{n_1 - s_1}(\gamma_1) > 0) \wedge ... \wedge (Rd_x^{n_r - s_r}(\gamma_r) > 0)])\}$$

pro $k = 0$, kde S je množina všech uspořádaných r-tic $(s_1, ..., s_r)$, $0 \leq s_1 \leq n_1$, ..., $0 \leq s_r \leq n_r$, a $\psi_{k,l} \equiv [(\gamma_{k,l+1} = 0) \wedge ... \wedge (\gamma_{k,n_k} = 0)]$ pro $0 \leq l < n_k$ a $\psi_{k,l} \equiv (0 = 0)$ pro $l = n_k$,

g) pro formuli Φ ve tvaru

$$(\exists_k x)\colon [(p = 0) \wedge (\gamma_1 > 0) \wedge ... \wedge (\gamma_r > 0)]$$

je

$$T(\Phi) \equiv \{[\neg(p_m = 0) \vee ... \vee \neg(p_0 = 0)] \wedge \Phi\} \vee$$
$$([(p_m = 0) \vee ... \vee (p_0 = 0)] \wedge$$
$$T\{(\exists_k x)\colon [(\gamma_1 > 0) \wedge ... \wedge (\gamma_r > 0)]\}),$$

h) pro formuli Φ ve tvaru

$$(\exists_k x)\colon [(\gamma_1 = 0) \wedge ... \wedge (\gamma_r = 0)]$$

je

$$T(\Phi) \equiv T\{(\exists_k x)\colon [P_x(\gamma_1^2 + ... + \gamma_r^2) = 0]\},$$

i) pro formuli Φ ve tvaru

$$(\exists_k x)\colon [(\gamma_1 = 0) \wedge ... \wedge (\gamma_s = 0) \wedge (\gamma_{s+1} > 0) \wedge ... \wedge (\gamma_r > 0)]$$

je

$$T(\Phi) \equiv T\{(\exists_k x)\colon [P_x(\gamma_1^2 + ... + \gamma_s^2) = 0 \wedge (\gamma_{s+1} > 0) \wedge ... \wedge (\gamma_r > 0)]\},$$

j) pro formuli Φ, která nepatří do žádné výše uvedené třídy, ale je obecně ve tvaru

$$\Phi \equiv (\exists_k x)\colon (\Phi_1 \wedge ... \wedge \Phi_r),$$

kde Φ_i je rovnost $\gamma_i = 0$ nebo nerovnost $\gamma_i > 0$, je

$$T(\Phi) \equiv T(\exists_k x)\colon (\Phi_{j_1} \wedge ... \wedge \Phi_{j_r}).\blacklozenge$$

Nakonec definujme operátor U.

Definice 3.4.13: Mějme tři libovolné formule Φ, Ψ a Θ v proměnné x. Potom pokud je:

a) formule Φ atomická formule, platí

$$U(\Phi) \equiv \Phi,$$

b) formule $\Phi \equiv (\Psi \vee \Theta)$, platí

$$U(\Phi) \equiv [U(\Psi) \vee U(\Theta)],$$

c) formule $\Phi \equiv (\Psi \wedge \Theta)$, platí

$$U(\Phi) \equiv [U(\Psi) \wedge U(\Theta)],$$

d) formule $\Phi \equiv \neg \Psi$, platí

$$U(\Phi) \equiv \neg U(\Psi),$$

e) formule $\Phi \equiv (\exists_k x)\colon \Psi$ a $QP_x U(\Psi) \equiv \Psi_1 \vee \ldots \vee \Psi_n$, kde Ψ_i jsou atomické formule, platí

$$U(\Phi) \equiv \neg T[(\exists_0 x)\colon \Psi_1] \vee \ldots \vee \neg T[(\exists_0 x)\colon \Psi_n]. \blacklozenge$$

Tím jsme dospěli k důsledku, že je-li Φ libovolná formule, potom $U(\Phi)$ je formule s ní ekvivalentní a navíc bez kvantifikátorů. Pomocí výše popsaného postupu je možné každou matematickou formuli přeformulovat ve formuli ekvivalentní a bez kvantifikátorů.

První část Tarského postupu je tak hotova, nyní ještě ve stručnosti naznačíme část druhou, v níž je u každé formule bez kvantifikátorů možné rozhodnout, je-li pravdivá, či nikoliv.

Tento postup v podstatě spočívá v prokázání, že daná formule je ekvivalentní s tvrzením $0 = 0$ anebo $0 = 1$.

Toho je docíleno tak, že každému výrazu p z formule je přiřazena hodnota $n(p)$ podle následujících zákonů:

a) platí

$$n(-1) = -1,\ n(0) = 0 \text{ a } n(1) = 1,$$

b) pokud $p = q + r$, pak platí

$$n(p) = n(q) + n(r),$$

pokud $p = q \cdot r$, pak platí

$$n(p) = n(q) \cdot n(r).$$

Zavedeme operátor W a definujme jej takto.

Definice 3.4.14: Nechť p, q jsou výrazy a Φ, Ψ a Θ formule bez kvantifikátorů. Potom platí:

a) je-li $\Phi \equiv (p = q)$, pak pro $n(p) = n(q)$ platí

$$W(\Phi) \equiv (0 = 0),$$

jinak

$$W(\Phi) \equiv (0 = 1),$$

b) je-li $\Phi \equiv (p > q)$, pak pro $n(p) > n(q)$ platí

$$W(\Phi) \equiv (0 = 0),$$

jinak

$$W(\Phi) \equiv (0 = 1),$$

c) je-li $\Phi \equiv (\Psi \vee \Theta)$, pak v případě, že $W(\Psi) \equiv (0 = 0)$ nebo $W(\Theta) \equiv (0 = 0)$, platí

$$W(\Phi) \equiv (0 = 0),$$

jinak

$$W(\Phi) \equiv (0 = 1),$$

d) je-li $\Phi \equiv (\Psi \wedge \Theta)$, pak v případě, že $W(\Psi) \equiv (0 = 0)$ a zároveň $W(\Theta) \equiv (0 = 0)$, platí

$$W(\Phi) \equiv (0 = 0),$$

jinak

$$W(\Phi) \equiv (0 = 1),$$

e) je-li $\Phi \equiv \neg \Psi$, pak pro $\Psi \equiv (0 = 0)$ platí

$$W(\Phi) \equiv (0 = 1),$$

jinak

$$W(\Phi) \equiv (0 = 0). \blacklozenge$$

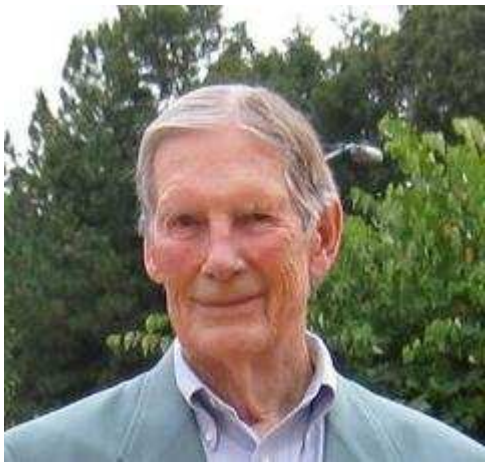
Z výše uvedeného vyplývá, že máme-li zadánu libovolnou matematickou formuli Φ s kvantifikátory, pak formule ve tvaru $WU(\Phi)$ je buď ve tvaru $0 = 0$ nebo $0 = 1$ a přitom je ekvivalentní s původní formulí Φ.

Pro každou formuli Φ je možné najít její ekvivalent $WU(\Phi)$ v konečném počtu kroků. Ten je dán tvarem, v němž je formule Φ zadána, především její délkou, množstvím kvantifikátorů a proměnných, které se v ní vyskytují, atd.

Z výše popsaného je jasné, jak komplikovaný celý Tarského postup eliminace kvantifikátorů byl, a nelze se tedy divit, že v době svého vzniku a publikování nenašel praktické uplatnění. [TAR57]

3.5 Nalezení algoritmu CAD

Ještě za Tarského života ale došlo k zásadnímu zlomu. S nástupem počítačů a s jejich rostoucím výkonem se objevila příležitost, kterou v sedmdesátých letech 20. století využil americký matematik George E. Collins, jenž s kolegy navrhl algoritmus cylindrické algebraické dekompozice umožňující efektivní provedení eliminace kvantifikátorů. Dosažené výsledky pak jeho tým publikoval v roce 1974 v článku *Quantifier elimination for real closed fields by cylindrical algebraic decomposition – preliminary report* v publikaci *SICSAM Bulletin* a o rok později v článku *Quantifier elimination for real closed fields by cylindrical algebraic decomposition* v *Proceedings Second GI Conference on Automata Theory and Formal Languages*.[8]



Obrázek 2b: George E. Collins v září 2009

První implementaci algoritmu CAD pak v letech 1979 a 1980 provedl Collinsův student a spolupracovník Dennis A. Arnon, v roce 1991 na toto navázal další Collinsův žák Hoon Hong podstatným vylepšením zavedením částečné cylindrické algebraické dekompozice. Na ní byl následně založen i program QEPCAD, kterému je věnována kapitola 6.3. [ACM84a, DAV09]

[8] Pro čtenáře je jejich práce v současné době asi nejlépe dostupná v článku *Cylindrical algebraic decomposition I: the basic algorithm* vydaném v *SIAM Journal on Computing* v roce 1982 a později ještě v roce 1984, na nějž pak navázali dalším textem s názvem *Cylindrical algebraic decomposition II: an adjacency algorithm for the plane* pojednávajícím o vylepšení metody CAD pro rozklad v rovině.

4 Jednodušší příklady středoškolské matematiky

Podívejme se nyní na několik příkladů, se kterými se studenti mohou setkat již na středních školách (ve všech úlohách se pohybujeme v oboru reálných čísel R, případně v prostorech vyšších dimenzí R^n):

Příklad 4.1: Nalezněte všechna reálná x taková, že platí nerovnost $x^2 \geq 0$.

V zadání se nachází známá matematická nerovnost, která říká, že druhá mocnina libovolného reálného čísla je větší než nula, popřípadě je rovna nule (tento případ nastane pro $x = 0$). Řešením jsou pak všechna reálná čísla, což můžeme zapsat jako

$$x \in R. \blacklozenge$$

Příklad 4.2: Nalezněte takové reálné číslo x, že pro libovolné reálné číslo y platí rovnost $x + y = 0$.

Jinými slovy máme podle zadání najít takové reálné číslo, které bude při sečtení s libovolným reálným prvkem dávat výsledek rovný 0. Takové číslo však neexistuje a tato rovnost platí pouze tehdy, když

$$x = -y.$$

Čísla x a y jsou tedy pevně svázána a takovou dvojici x a y, jak je požadováno v zadání, nedokážeme nalézt. Zapišme výsledek

$$x \in \{\}. \blacklozenge$$

Příklad 4.3: Ukažte, že existuje reálné číslo x, pro které platí rovnost $x \cdot y = 0$, y je libovolné reálné číslo.

Na rozdíl od předchozího příkladu, pro operaci násobení existuje číslo x, které v součinu s libovolným reálným číslem dá výsledek roven 0, a tím je číslo 0, které je agresivním prvkem operace násobení. Výsledek tak můžeme zapsat

$$x = 0. \blacklozenge$$

Příklad 4.4: Určete reálné číslo x, které splňuje rovnost $|x - 1| + |x + 1| = 2$.[9]

Podle zadání máme nalézt číslo x takové, že součet absolutních hodnot jeho součtů s čísly 1 a –1 je roven 2. Pro jednodušší představu přeformulujme zadání do této

[9] V tomto motivačním příkladu upozorněme na skutečnost, že jeho řešení prostřednictvím cylindrické algebraické dekompozice není kvůli absolutním hodnotám tak úplně jednoduché, například program QEPCAD by s jeho zadáním měl problémy. Celkem však jde tato nepříjemnost obejít odstraněním absolutních hodnot metodou rozdělení základní množiny, přičemž je pak již možné na jejích jednotlivých intervalech úlohu dořešit.

podoby: Existuje takové číslo x, jehož součet vzdáleností na číselné ose od čísel 1 a –1 je roven celkové vzdálenosti 2. Vzhledem k tomu, že sama vzdálenost mezi čísly 1 a –1 je rovna 2, platí tato rovnost pro libovolné číslo z intervalu

$$\langle -1; 1\rangle. \blacklozenge$$

Poznámka: Z hlediska středoškolské matematiky je Příklad 4.4 poměrně zajímavý, výsledkem rovnice je totiž v tomto případě interval reálných čísel, nikoliv množina izolovaných bodů, jak tomu zpravidla bývá u příkladů, se kterými se studenti setkávají. ♦

Tyto příklady nejsou nikterak obtížné, studenti je vesměs bez větších obtíží dokážou vyřešit, ale aniž by si to sami uvědomovali, pohybují se na okraji oblasti eliminace kvantifikátorů. Všechny výše jmenované úlohy a jim typově podobné příklady jdou velmi jednoduše přetransformovat do kvantifikovaných formulí obsahujících polynomiální rovnice a nerovnice.

Hledáme-li kupříkladu v Příkladu 3.1 reálná čísla x, jejichž druhá mocnina je nezáporným číslem, lze problém popsat kvantifikovanou formulí

$$(\exists x)\colon x^2 \geq 0 \text{ nebo } (\forall x)\colon x^2 \geq 0.$$

V prvním případě tvrdíme, že existuje alespoň jedno reálné číslo x, pro které platí zadaná nerovnice, ve druhém případě říkáme, že tato vlastnost platí pro všechna reálná čísla x. Ekvivalentními formulemi, tentokrát již bez kvantifikátorů, pak jsou zápisy

$$0 = 0,$$

počítačový program nás též může potěšit výsledkem *True*. Takový výsledek interpretujeme jako souhlasnou odpověď. V případě formule

$$(\exists x)\colon x^2 \geq 0$$

se dozvídáme, že alespoň jedno takové reálné číslo existuje (nic bližšího však už nová formule bez kvantifikátorů neříká), naproti tomu v případě formule

$$(\forall x)\colon x^2 \geq 0$$

se potvrdí správnost tvrzení, že skutečně všechna reálná čísla x mají požadovanou vlastnost.

Obdobně v ostatních případech je možné zadání přepsat do kvantifikovaných formulí

$$(\exists x)\colon x + y = 0,\ (\exists x)\colon x \cdot y = 0 \text{ a } (\exists x)\colon |x - 1| + |x + 1| = 2.$$

4.1 Řešení jednoduchých příkladů rozkladem podle nulových bodů

Řešení předešlých úloh byla poměrně jednoduchá, k výsledku šlo dospět jednoduchou logickou úvahou, většinou se ale setkáme se zadáními představujícími určitý matematický problém (například hledání kořenů polynomiálních rovnic a nerovnic), kdy už s pouhou úvahou nevystačíme.

Příklad 4.1.1: Určete čísla $x \in R$, pro která platí nerovnice $x^4 + 2x^3 - 21x^2 - 22x + 40 < 0$.

Tento příklad je již obtížnější, po řešiteli požaduje nejprve rozklad polynomu na levé straně nerovnice a následně s pomocí takto získaných reálných kořenů zjištění intervalů, v nichž je funkční hodnota polynomu záporná. Celý problém jde zároveň zapsat jako kvantifikovanou formuli ve tvaru

$$(\exists x)\colon x^4 + 2x^3 - 21x^2 - 22x + 40 < 0$$

a řešit jej metodou cylindrické algebraické dekompozice, která se však v podstatě shoduje s postupem řešení nerovnice

$$x^4 + 2x^3 - 21x^2 - 22x + 40 < 0$$

pomocí metody nulových bodů, se kterou jsou studenti středních škol již seznámeni. Prvním krokem je tedy nalezení kořenů polynomu

$$p(x) = x^4 + 2x^3 - 21x^2 - 22x + 40.$$

Těmi jsou postupně body –5; –2; 1 a 4, jelikož polynom $p(x)$ lze zapsat jako součin faktorů

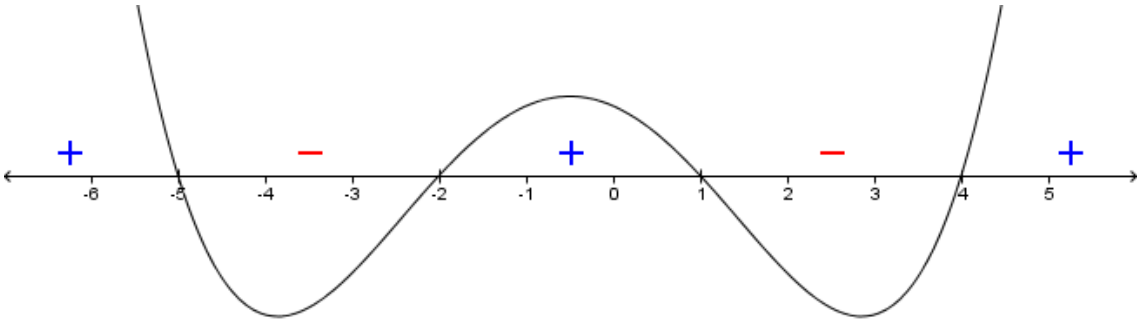
$$p(x) = (x + 5)(x + 2)(x - 1)(x - 4).$$

Tato čtveřice kořenů rozdělila číselnou osu na pětici intervalů

$$(-\infty; -5),\ (-5; -2),\ (-2; 1),\ (1; 4) \text{ a } (4; \infty),$$

v nichž si polynom zachovává stejné znaménko (v každém z intervalů je hodnota polynomu $p(x)$ buď kladná anebo záporná). Přitom pro všechny hodnoty z intervalu $(-\infty; -5)$ platí, že jednotliví činitele součinu $(x + 5)(x + 2)(x - 1)(x - 4)$ jsou záporní, zároveň je jich sudý počet, takže výsledná hodnota je pro celý interval kladná. Pro interval $(-5; -2)$ jsou všichni činitele záporní až na závorku $(x + 5)$, která je kladná. Protože se jedná o součin tří záporných čísel a jednoho kladného, je výsledná hodnota součinu záporná. Obdobným způsobem dostaneme výsledky i pro

zbývající intervaly, přičemž záporné výsledky nalezneme v intervalech $(-5; -2)$ a $(1; 4)$, ve zbývajících třech intervalech jsou hodnoty polynomu kladné.



Obrázek 3: Číselná osa rozdělená na části, v nichž funkční hodnota nabývá kladných a záporných hodnot

Podařilo se nalézt takové hodnoty proměnné x, které vyhovují zadané formuli, je jimi sjednocení otevřených intervalů

$$(-5; -2) \cup (1; 4)$$

(případně můžeme k vyjádření oboru pravdivosti použít zápis soustavy nerovnic $[x > -5 \wedge x < -2] \vee [x > 1 \wedge x < 4]$). ♦

Příklad 4.1.2: Ověřte pravdivost formule $(\forall y)$: $y^3 - 3y^2 + 2y - 6 \geq 0$.

Nyní je zadání zapsáno přímo kvantifikovanou formulí, přičemž výsledek úlohy odpovídá tomu, zda má nerovnice

$$y^3 - 3y^2 + 2y - 6 \geq 0$$

řešení ve všech prvcích y množiny reálných čísel, či nikoliv.

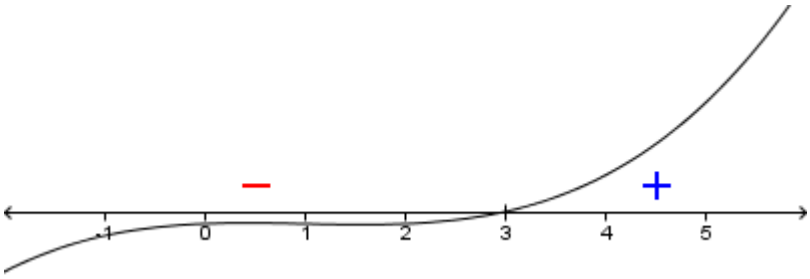
Provedeme opět rozklad polynomu, přičemž díky lichému stupni polynomu je zřejmé, že bude mít alespoň jeden reálný kořen,

$$p(y) = y^3 - 3y^2 + 2y - 6$$

a dostaneme

$$p(y) = (y - 3)(y^2 + 2).$$

Polynom má nyní pouze jeden reálný kořen s hodnotou $y_0 = 3$, který je zároveň hledaným nulovým bodem, zbývající dva komplexní kořeny $\pm\sqrt{2}i$ nás nezajímají, neboť pracujeme nad tělesem reálných čísel. Tento nulový bod rozdělí číselnou osy na dva intervaly $(-\infty; 3)$ a $(3; \infty)$. Pro libovolnou hodnotu z intervalu $(-\infty; 3)$ nabývá polynom záporné funkční hodnoty, pro libovolné číslo z intervalu $(3; \infty)$ dostáváme výsledek kladný a pro samotný nulový bod $y_0 = 3$ máme rovnost $p(y) = 0$.



Obrázek 4: Číselná osa rozdělená na části,
kde funkční hodnota nabývá kladné a záporné hodnoty

Podle zadání má nerovnost platit pro všechna $y \in R$. Jak ale vidíme z výše uvedeného, nerovnost platí pouze pro hodnoty

$$y \geq 3,$$

pro $y < 3$ tomu tak již není. Formule

$$(\forall y)\colon y^3 - 3y^2 + 2y - 6 \geq 0$$

je tedy nesplnitelná formule a její ekvivalentní zápis bez kvantifikátorů je *False*.♦

Poznámka: Metodou nulových bodů lze přirozeně řešit i Příklady 4.1. a 4.4. V případě nerovnice

$$x^2 \geq 0$$

je to velice jednoduché, existuje pouze jeden nulový bod, kterým je $x_0 = 0$ a který rozděluje množinu všech reálných čísel do dvou intervalů $(-\infty; 0)$ a $(0; \infty)$. V obou těchto intervalech jsou však hodnoty mnohočlenu x^2 kladné a splňují zadání. Pak již zbývá ověřit ještě nulový bod $x_0 = 0$, který též vyhovuje, a oborem pravdivosti je celá množina všech reálných čísel.

Při řešení rovnosti s absolutními hodnotami

$$|x - 1| + |x + 1| = 2$$

jde již o trochu složitější problém. Nejprve je nutné určit nulové body pro obě absolutní hodnoty, těmi jsou $x_0 = \pm 1$, které opět rozdělí číselnou osu na disjunktní intervaly $(-\infty; -1)$, $(-1; 1)$ a $(1; \infty)$. Ve všech těchto intervalech odstraníme absolutní hodnoty a dostaneme:

a) pro $x \in (-\infty; -1)$:

$$-(x - 1) - (x + 1) = 2$$
$$-x + 1 - x - 1 = 2$$
$$-2x = 2$$
$$x = -1$$

Hodnota $x = -1$ je ovšem nulový bod a nepatří do intervalu $(-\infty; -1)$, takže pravdivostní obor P je zatím prázdná množina.

b) pro $x \in (-1; 1)$:

$$-(x-1) + (x+1) = 2$$
$$-x + 1 + x + 1 = 2$$
$$0 = 0$$

Rovnost $0 = 0$ platí pro všechna x z intervalu $(-1; 1)$, a proto $P = (-1; 1)$.

c) pro $x \in (1; \infty)$:

$$(x-1) + (x+1) = 2$$
$$2x = 2$$
$$x = 1$$

Hodnota $x = 1$ je druhý nulový bod a nepatří do intervalu $(1; \infty)$, to znamená, že jsme nenalezli žádná nová čísla x vyhovující rovnici.

Nakonec zbývá ověřit ještě nulové body $x_0 = \pm 1$, které můžeme do rovnice rovnou dosadit, přičemž pro oba vyjde rovnost $0 = 0$, tudíž oba nulové body patří do oboru pravdivosti

$$P = \langle -1; 1 \rangle.$$

Vidíme, že jsme se dobrali stejného výsledku jako v předchozím postupu.♦

5 Základní myšlenka cylindrické algebraické dekompozice

Základní idea CAD se nejlépe demonstruje na jednoduchých formulích, které mohou vycházet například ze soustav rovnic a nerovnic. Ty lze totiž poměrně jednoduše převést na formule obsahující kvantifikátory a s nimi dále pracovat.

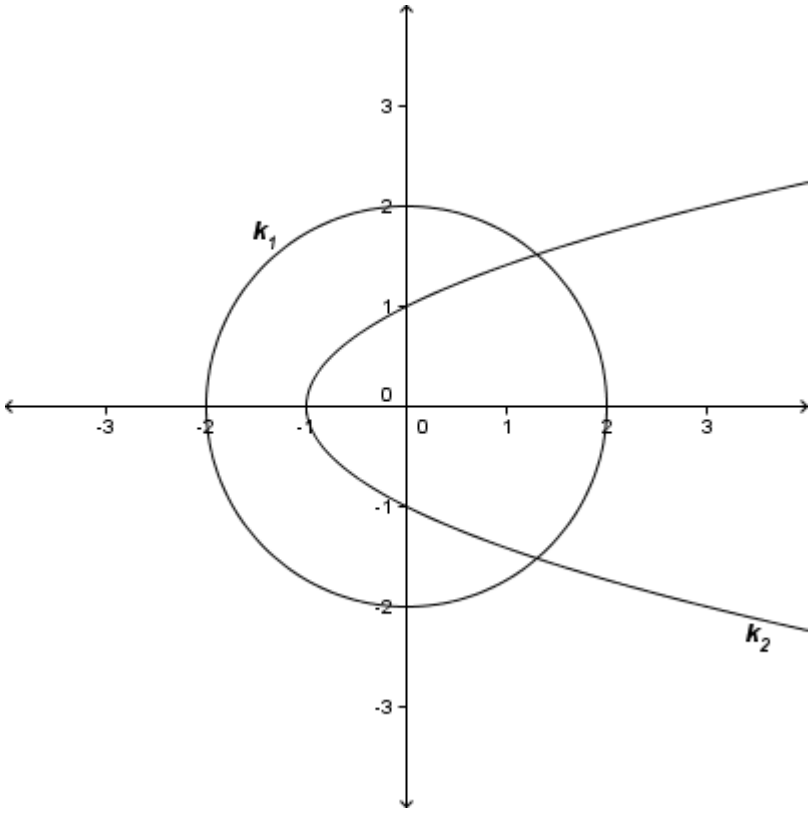
Příklad 5.1: Proveďte eliminaci kvantifikátorů metodou cylindrické algebraické dekompozice ve formuli $(\exists x)$: $x^2 + y^2 - 4 \leq 0 \wedge x - y^2 + 1 < 0$.

Soustavu dvou nerovnic o dvou neznámých

$$x^2 + y^2 - 4 \leq 0$$
$$x - y^2 + 1 < 0,$$

která odpovídá zadané kvantifikované formuli, jde samozřejmě řešit i jinými přístupy, to nám však nebrání ji použít k popisu řešení prostřednictvím CAD, jenž povede k nalezení kvantifikátory neobsahující výstupní formule ekvivalentní s formulí zadanou. Celý příklad doprovodíme ilustračními obrázky.

Jak můžeme vidět na následujícím obrázku, omezujícími křivkami jsou v tomto případě k_1: $x^2 + y^2 - 4 = 0$ (kružnice zadaná středovou rovnicí $x^2 + y^2 = 4$ se středem v bodě $S = [0; 0]$ a poloměrem $r = 2$) a k_2: $x - y^2 + 1 = 0$ (parabola s vrcholem $V = [-1; 0]$, ohniskem $F = \left[-\frac{3}{4}; 0\right]$ a řídící přímkou $d : x = -\frac{5}{4}$).



Obrázek 5: Grafické znázornění zadané situace

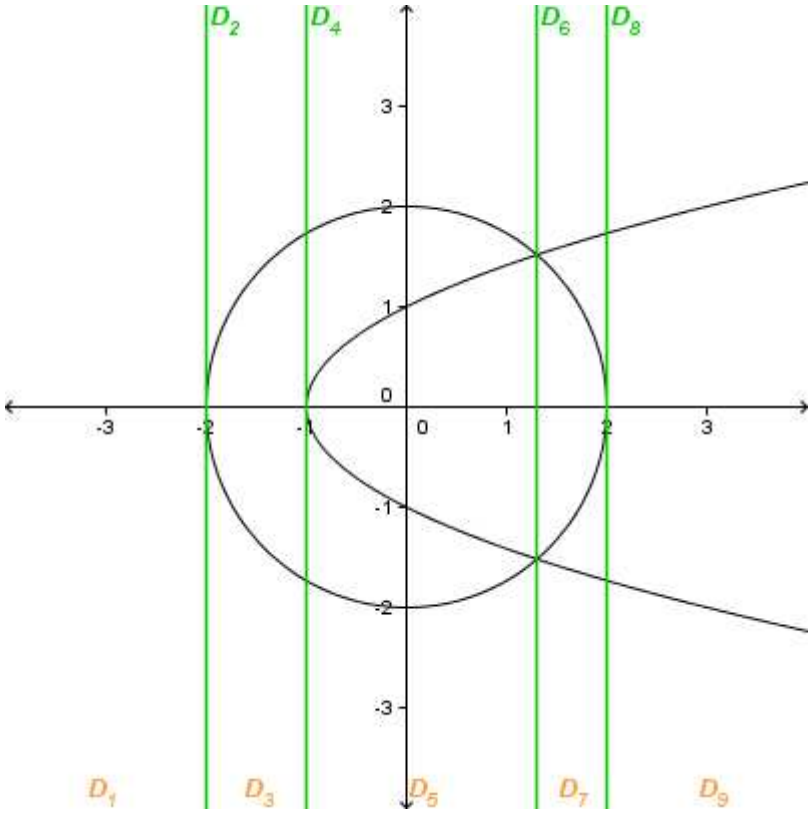
Základní myšlenkou je rozdělení *r*-rozměrného prostoru (v tomto případě R^2) na konečný počet disjunktních oblastí, tzv. buněk (cells), ve kterých pomocí vhodně zvolených bodů – vzorků (samples) jednoduše tvrzení ověříme. Přitom je ovšem nutné zajistit, aby pro každou jednotlivou buňku rozkladu platilo, že ve všech jejích bodech je zachováno stejné znaménko funkční hodnoty u všech polynomů. Obecně není tato podmínka nikterak samozřejmá, ale u relativně jednoduchých polynomiálních výrazů (relativně jednoduchých v porovnání s nejrůznějšími logaritmickými, exponenciálními či goniometrickými výrazy), kterými se budeme zabývat, je tato podmínka naplněna.

V takovém případě mluvíme o tom, že daný rozklad *D* soustavy rovnic a nerovnic *A* je *A*-invariantní, tedy že na jednotlivých intervalech zachovávají výrazy své kladné, záporné anebo nulové hodnoty.

Nejprve provedeme dekompozici *D* prostoru R^2 podle jedné proměnné; v tomto případě začněme například proměnnou *x*, obecně ale může být pořadí proměnných i jiné. Do rozkládaného prostoru zavedeme přímky kolmé k ose *x* procházející nulovými body a body obratů zadaných polynomů a jejich společnými průsečíky. Tak dostaneme rozklad

$$D = (D_1; D_2; ...; D_9),$$

přičemž $D_1 = (-\infty; -2)$; $D_2 = \{-2\}$; $D_3 = (-2; -1)$; $D_4 = \{-1\}$; $D_5 = (-1; \alpha)$; $D_6 = \{\alpha\}$; $D_7 = (\alpha; 2)$; $D_8 = \{2\}$; $D_9 = (2; \infty)$, $\alpha = -\frac{1}{2} + \frac{\sqrt{13}}{2}$. Oblasti D_1, D_3, D_5, D_7 a D_9 jsou tzv. sektory (sectors) na R^2 a oblasti D_2, D_4, D_6 a D_8 se nazývají sekce (sections) na R^2, dohromady je nazveme cylindry nad příslušnou oblastí a budeme je značit $C(D_1)$, $C(D_2)$, atd.



Obrázek 6: Rozklad D prostoru R^2
(sektory jsou označeny oranžově, sekce zeleně)

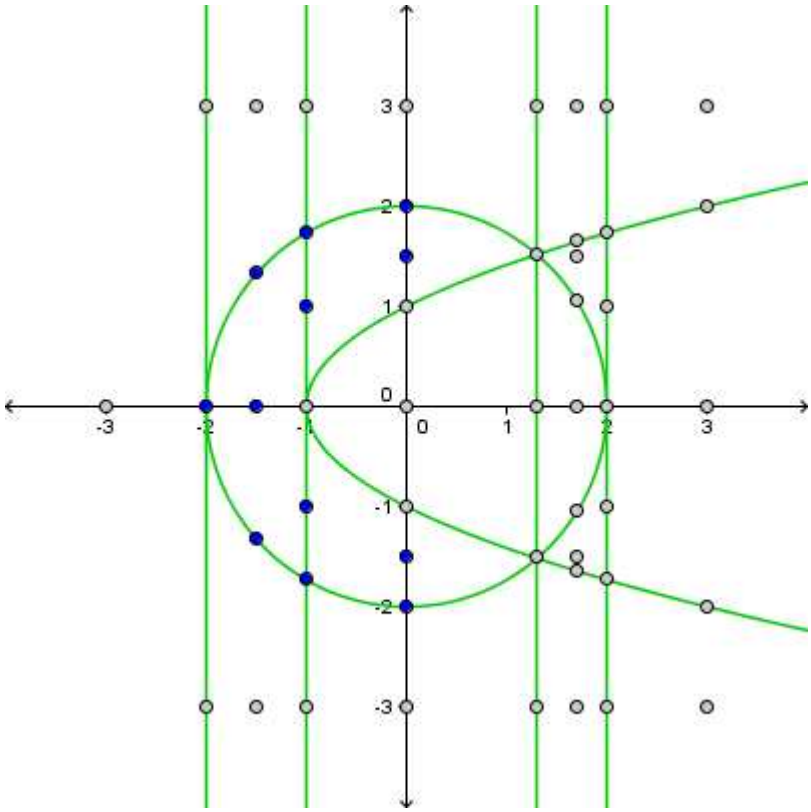
V každé z oblastí D_1 až D_9 následně provedeme další rozklad podle zobrazených křivek k_1, k_2 a jejich významných bodů. Potom například v oblasti D_1 bude rozklad velmi jednoduchý, neboť se zde nenalézá žádná z křivek a můžeme tedy psát $D_1 = R$. Naproti tomu v oblastech D_4 nebo D_5 bude rozklad poněkud komplikovanější, jelikož se zde nachází obě křivky k_1 a k_2. V sekci D_4 se „pohybujeme“ po jednorozměrné přímce, jejíž rozklad provedeme právě pomocí křivek k_1 a k_2, a dostaneme $D_4 = (D_{4,1}; D_{4,2}; ...; D_{4,7})$, přičemž oblasti $D_{4,1}$, $D_{4,3}$, $D_{4,5}$ a $D_{4,7}$ nazveme sektory na D_4 a oblasti $D_{4,2}$, $D_{4,4}$ a $D_{4,6}$ jsou sekcemi na D_4. Obdobně postupujeme při rozkladu D_5, tentokrát jde o část roviny, kterou opět rozdělíme podle křivek k_1 a k_2. Dostaneme rozklad $D_5 = (D_{5,1}; D_{5,2}; ...; D_{5,9})$, kde analogicky budou oblasti $D_{5,1}$, $D_{5,3}$, $D_{5,5}$, $D_{5,7}$ a $D_{5,9}$ nazývány sektory na D_5, oblasti $D_{5,2}$, $D_{5,4}$,

$D_{5,6}$ a $D_{5,8}$ jsou sekce na D_5. Stejným způsobem provedeme i zbývající rozklady a obdržíme množinu disjunktních buněk, které tvoří rozklad $D = (D_1; D_{2,1}; D_{2,2}; D_{2,3}; D_{3,1}; ...; D_{9,4}; D_{9,5})$ prostoru R^2.

Nyní v každé z takto vzniklých buněk určíme testovací bod a sestavíme z nich cylindrický algebraický vzorek (cas; cylindric algebraic sample). Vyberme vzorek rozkladu D například takto:

$$
\begin{aligned}
s = (&[-3;0],\\
&[-2;-3],[-2;0],[-2;3],\\
&\left[-\tfrac{3}{2};-3\right],\left[-\tfrac{3}{2};-\tfrac{\sqrt{7}}{2}\right],\left[-\tfrac{3}{2};0\right],\left[-\tfrac{3}{2};\tfrac{\sqrt{7}}{2}\right],\left[-\tfrac{3}{2};3\right],\\
&[-1;-3],\left[-1;-\sqrt{3}\right],[-1;-1],[-1;0],[-1;1],\left[-1;\sqrt{3}\right],[-1;3],\\
&[0;-3],[0;-2],\left[0;-\tfrac{3}{2}\right],[0;-1],[0;0],[0;1],\left[0;\tfrac{3}{2}\right],[0;2],[0;3],\\
&[\alpha;-3],[\alpha;-\beta],[\alpha;0],[\alpha;\beta],[\alpha;3],\\
&[1{,}7;-3],\left[1{,}7;-\sqrt{2{,}7}\right],\left[1{,}7;-\tfrac{3}{2}\right],\left[1{,}7;-\sqrt{1{,}11}\right],[1{,}7;0],\left[1{,}7;\sqrt{1{,}11}\right],\left[1{,}7;\tfrac{3}{2}\right],\left[1{,}7;\sqrt{2{,}7}\right],\\
&[1{,}7;3],\\
&[2;-3],\left[2;-\sqrt{3}\right],[2;-1],[2;0],[2;1],\left[2;\sqrt{3}\right],[2;3],\\
&[3;-3],[3;-2],[3;0],[3;2],[3;3]),
\end{aligned}
$$

kde $\alpha = \dfrac{-1+\sqrt{13}}{2}$ a $\beta = \sqrt{\dfrac{1+\sqrt{13}}{2}}$.



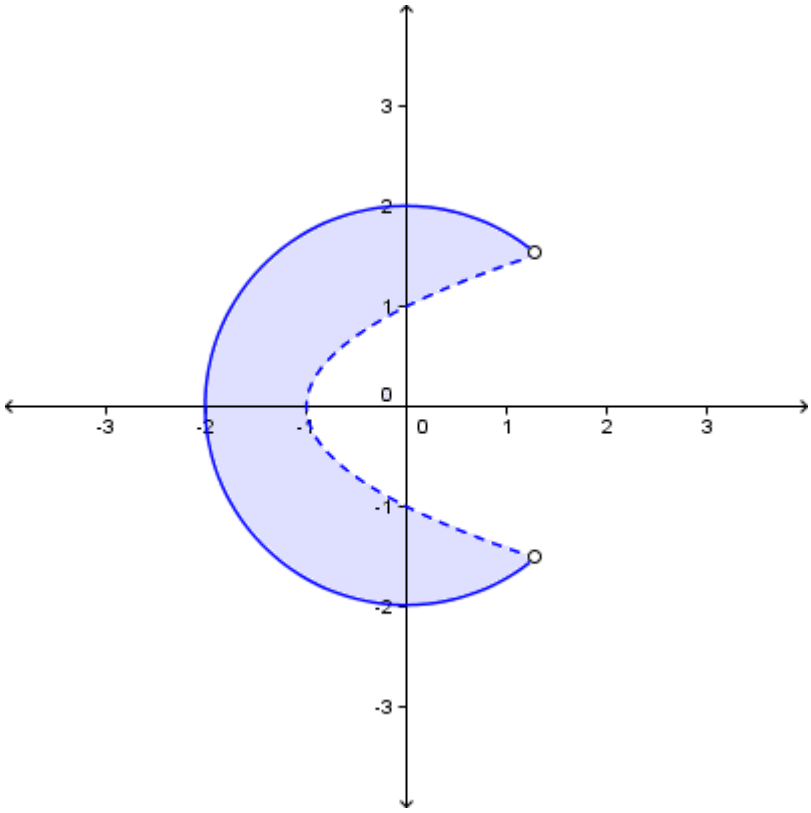
Obrázek 7: Vybrané vzorky jednotlivých buněk
(vyhovující vzorky jsou modré, nevyhovující šedé)

Využijeme toho, že pro celý soubor testovacích bodů platí, že je *s*-invariantní, pak již stačí jen u každého z těchto testovacích bodů ověřit, zda splňuje nerovnosti

$$x^2 + y^2 - 4 \leq 0 \text{ a } x - y^2 + 1 < 0$$

ze zadané formule. Přitom víme, že vyhovuje-li daný vzorek zadaným polynomiálním nerovnicím, potom jim vyhovuje i celá buňka, do níž tento bod náleží.

Konečným výsledkem, který je řešením zadaných rovnic a nerovnic, resp. kvantifikované formule, je sjednocení všech vyhovujících oblastí.



Obrázek 8: Sjednocení buněk vyhovujících zadané formuli (včetně části křivky k_1)

Námi hledané reálné číslo x tedy skutečně existuje a může nabývat hodnot určených právě získaným výsledkem, jde o polouzavřený interval

$$\left\langle -2; \frac{-1+\sqrt{13}}{2} \right).$$

K zápisu nové formule, tentokráte již bez kvantifikátorů, ekvivalentní s původní kvantifikovanou formulí využijeme proměnnou y, pomocí jejíchž hodnot určíme ty případy, kdy zadaná formule platí. Píšeme

$$(\exists x)\colon x^2 + y^2 - 4 \leq 0 \wedge x - y^2 + 1 < 0 \Leftrightarrow -2 \leq y \in \leq 2.\blacklozenge$$

Příklad 5.2: Proveďte eliminaci kvantifikátorů metodou cylindrické algebraické dekompozice ve formuli $(\exists y)$: $9x^2 + 4y^2 - 18x \leq 0 \wedge x^3 - y^2 - 4x = 0$.

Tento příklad je o něco složitější, neboť v zadané formuli se kromě nerovnice

$$9x^2 + 4y^2 - 18x \leq 0$$

vyskytuje i rovnost

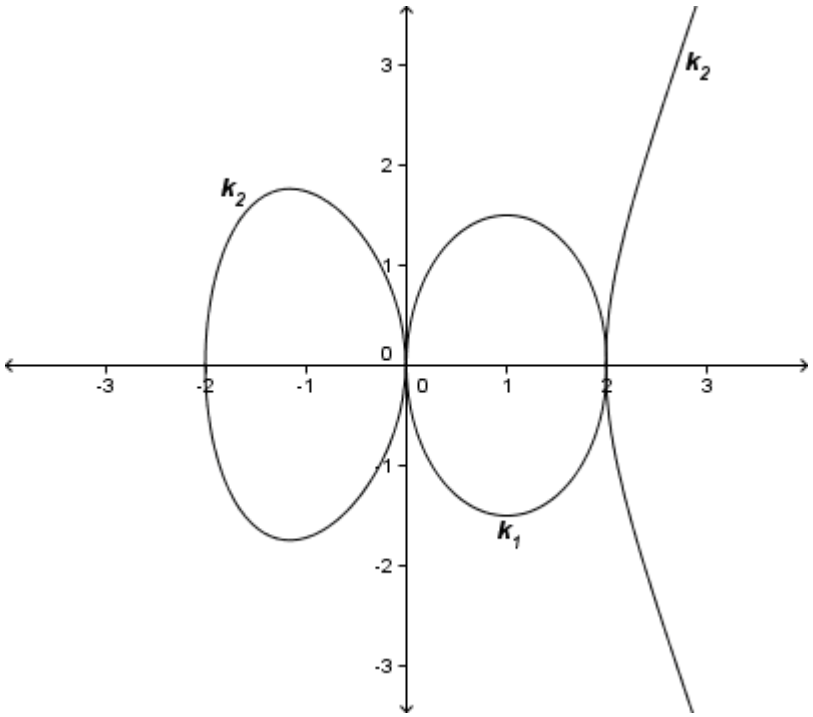
$$x^3 - y^2 - 4x = 0$$

obsahující vyšší mocniny proměnných. To však není překážkou při provedení cylindrické algebraické dekompozice.

Podívejme se pro úplnost nejprve na zadané křivky. Zápis křivky k_1: $9x^2 + 4y^2 - 18x = 0$ představuje elipsu (můžeme jej převést do přehlednějšího tvaru $\frac{(x-1)^2}{1} + \frac{y^2}{9/4} = 1$) se středem $S = [1; 0]$ a délkami hlavní poloosy $a = \frac{3}{2}$ a vedlejší poloosy $b = 1$. Naproti tomu křivka vyššího řádu k_2: $x^3 - y^2 - 4x = 0$ je již hůře představitelná, ale čistě pro představu ji můžeme bez problému nahradit křivkami dvojice funkcí

$$f_{1,2} = \pm\sqrt{x^3 - 4x}\,.$$

Nastalou situaci reprezentuje následující obrázek.

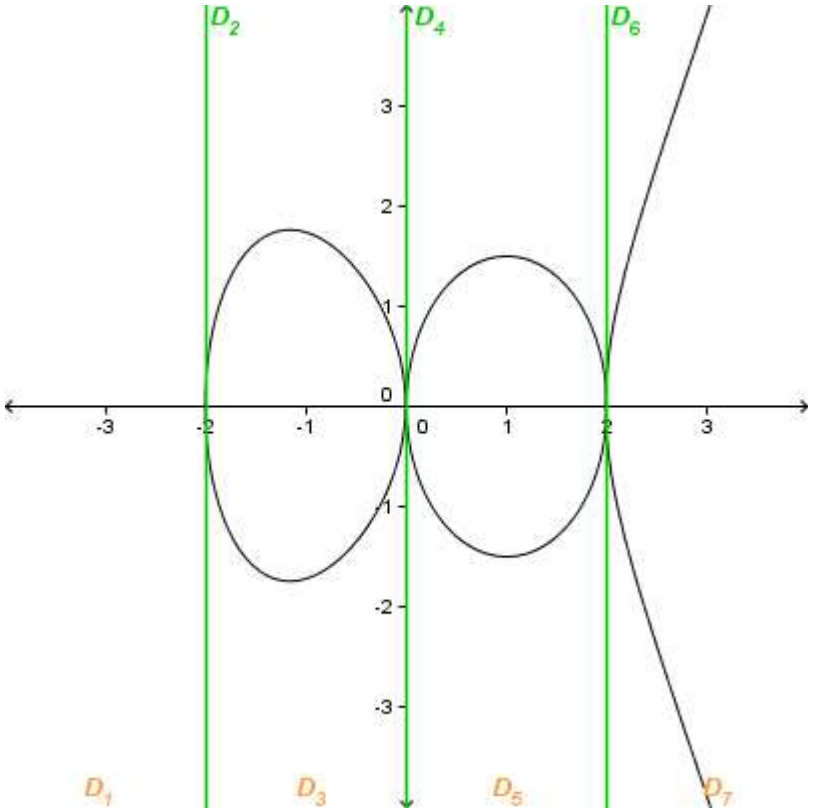


Obrázek 9: Křivky k_1 a k_2 v prostoru R^2

Započněme s rozkladem D prostoru R^2. Opět sestrojíme tečny k zadaným křivkám a přímky procházející průsečíky křivek, přičemž takto vzniklé přímky jsou kolmé k ose x. Tím dostaneme rozklad

$$D = (D_1; D_2; ...; D_7)$$

a platí $D_1 = (-\infty; -2)$, $D_2 = \{-2\}$, $D_3 = (-2; 0)$, $D_4 = \{0\}$, $D_5 = (0; 2)$, $D_6 = \{2\}$, $D_7 = (2; \infty)$. Oblasti rozkladu D můžeme jako v předchozím příkladu rozdělit na sektory D_1, D_3, D_5, D_7 a sekce D_2, D_4, D_6 na R^2.



Obrázek 10: Rozklad D prostoru R^2
(sektory jsou označeny oranžově, sekce zeleně)

Nyní provedeme rozklad jednotlivých oblastí D_1 až D_7 podle zadaných křivek k_1 a k_2, čímž obdržíme konečný počet buněk rozkladu

$$D = (D_1; D_{2,1}; D_{2,2}; ...; D_{7,4}; D_{7,5})$$

potřebných ke zvolení bodů testovacího vzorku a ověření kvantifikátory obsahující formule

$$(\exists y)\colon 9x^2 + 4y^2 - 18x \leq 0 \wedge x^3 - y^2 - 4x = 0.$$

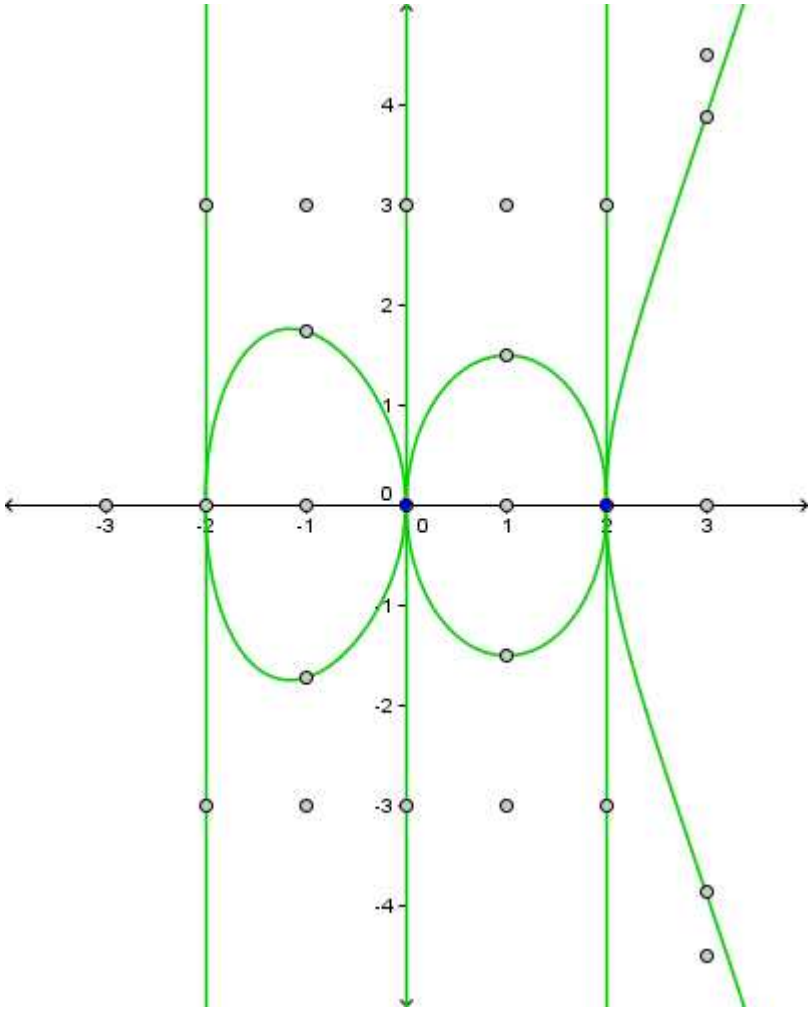
Zvolme testovací body rozkladu D například takto:

$$\begin{aligned} s = (&[-3;0],\\ &[-2;-3],[-2;0],[-2;3],\\ &[-1;-3],[-1;-\sqrt{3}],[-1;0],[-1;\sqrt{3}],[-1;3],\\ &[0;-3],[0;0],[0;3],\\ &[1;-3],\left[1;-\frac{3}{2}\right],[1;0],\left[1;\frac{3}{2}\right],[1;3],\\ &[2;-3],[2;0],[2;3],\\ &\left[3;-\frac{9}{2}\right],[3;-\sqrt{15}],[3;0],[3;\sqrt{15}],\left[3;\frac{9}{2}\right]). \end{aligned}$$

Po dosazení je pak již jasné, že nerovnostem

$$9x^2 + 4y^2 - 18x \leq 0,\ x^3 - y^2 - 4x = 0$$

vyhovují pouze vzorky [0; 0] a [2; 0] odpovídající buňkám $D_{4,2}$ a $D_{6,2}$.



Obrázek 11: Vybrané vzorky jednotlivých buněk (vyhovující vzorky jsou modré)

Vzhledem k tomu, že jde o izolované body, bude i řešením soustavy dvojice bodů [0; 0] a [2; 0]. Případy, kdy zadaná kvantifikovaná formule

$$(\exists y)\colon 9x^2 + 4y^2 - 18x \leq 0 \wedge x^3 - y^2 - 4x = 0$$

platí, zapíšeme pomocí proměnné x a získáme tak ekvivalentní formuli bez kvantifikátorů

$$x = 0 \vee x = 2.\blacklozenge$$

Vnímavý čtenář si jistě povšimne toho, že podle rozkladu nad daným prostorem R^n lze v cylindrické algebraické dekompozici zkonstruovat stromovou strukturu, jejímž kořenem je celý rozklad D. Ten je rodičem pro jednotlivé sekce a sektory D_1, D_2, ..., D_m, které jsou rodiči dalším uzlům. V úrovni listů této datové struktury pak figurují jednotlivé buňky CAD. Tohoto poznatku se využívá ve druhé fázi algoritmizované CAD, jak uvidíme v další kapitole.

6 Algoritmizace cylindrické algebraické dekompozice

Jako u naprosté většiny matematických (nebo alespoň částečně matematických) problémů je vhodné, když se dá k řešení použít nějaký algoritmický přístup. Pokud totiž žádný algoritmický postup neexistuje, může se celý problém ukázat jako velice složitý, či dokonce neřešitelný.

To je i případ eliminace kvantifikátorů, která, jak již bylo řečeno v úvodu, byla v počátcích svého vývoje pokládána za příliš komplikovanou a i sám Alfred Tarski předpokládal, že právě kvůli její složitosti nebude efektivně využitelná a navrhoval dokonce bádání na tomto matematicko-logickém poli vědy zanechat.

Teprve až s nástupem rychlejších výpočetních prostředků ve druhé polovině 20. století, vhodného softwarového vybavení a především cylindrické algebraické dekompozice se věci úspěšně pohnuly kupředu.

Celý výše popsaný postup se zdá být poměrně lehkým, avšak to platí v podstatě pouze v případě, že jej pojímáme „intuitivně“. Ve skutečnosti je i nadále eliminace kvantifikátorů dosti náročná (hlavně paměťově, provádíme-li ji ve vícerozměrném prostoru R^n) a od prvního publikování ideje CAD bylo provedeno několik kroků k optimalizování jejího průběhu.

Podívejme se, jak algoritmus pro eliminaci kvantifikátorů prostřednictvím CAD vypadá. Skládá se ze tří samostatných fází, kterými jsou projekční fáze (projection phase; v této fázi je ze vstupních dat konstruována tzv. projekční množina, která popisuje množinu polynomů vstupní formule), konstrukce hald (base phase; na základě projekční fáze je vytvořena množina vzorků jednotlivých buněk vzniklých v CAD) a konstrukce výstupní formule (extension phase; podle pravdivostních hodnot jednotlivých buněk CAD je konstruována výstupní formule). [ACM84a, DAV09, WIN96]

6.1 Projekční fáze

V první fázi – projekční – se vytvoří popis cylindrické algebraické dekompozice v prostoru všech proměnných vyskytujících se ve vstupní formuli. K tomu potřebujeme definovat několik předpisů a proměnných:

Definice 6.1.1: Mějme polynom

$$p \in R[x_1, x_2, ..., x_{k-1}][x_k]$$

daný předpisem

$$p = \sum_{i=1}^{n} p_i(x_1, x_2, \ldots, x_{k-1}) \cdot x_k^i \ .$$

Pak stupeň polynomu p značíme $\deg(p) = n$, vedoucí koeficient $\mathrm{lc}(p) = p_i$ a vedoucí člen polynomu $\mathrm{lt}(p) = p_n \cdot x_k^n$. ♦

Definice 6.1.2: Reduktum polynomu p je dáno vztahem

$$\mathrm{red}(p) = p - \mathrm{lt}(p),$$

k-té reduktum definujeme jako

$$\mathrm{red}^k(p) = \mathrm{red}(\mathrm{red}^{k-1}(p)) \text{ pro } k \geq 1,$$

přičemž nulté reduktum

$$\mathrm{red}^0(p) = p. \blacklozenge$$

Dále definujme Sylvestrovu matici, kterou budeme také potřebovat:

Definice 6.1.3: Nechť jsou dány dva polynomy v proměnné x_k

$$p = \sum_{i=1}^{n} p_i \cdot x_k^i \ , \ q = \sum_{j=1}^{m} q_j \cdot x_k^j \ , \text{ kde } p_n \neq 0, q_m \neq 0.$$

Potom čtvercová matice řádu $m + n$

$$S_{p,q} = \begin{pmatrix} p_n & p_{n-1} & \cdots & p_1 & p_0 & & & \\ & p_n & p_{n-1} & \cdots & p_1 & p_0 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & p_n & p_{n-1} & \cdots & p_1 & p_0 \\ q_m & q_{m-1} & \cdots & q_1 & q_0 & & & \\ & q_m & q_{m-1} & \cdots & q_1 & q_0 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & q_m & q_{m-1} & \cdots & q_1 & q_0 \end{pmatrix}$$

se nazývá Sylvestrovou maticí (na prázdných ve schématu nevypsaných místech se vyskytují nuly). Determinant Sylvestrovy matice se nazývá Sylvestrův rezultant a je dán vztahem

$$\mathrm{res}(p, q) = \det(S_{p,q}).$$

Dále definujme základní subrezultant

$$\mathrm{psc}_l(p, q) = \det(S_{p,q}^l),$$

kde matice $S_{p,q}^l$ vznikne z matice $S_{p,q}$ ostraněním posledních $2l$ sloupců a řádků $n - l + 1, n - l + 2, \ldots, n - 1, n$ a $n + m - l + 1, n + m - l + 2, \ldots, n + m - 1, n + m$. ♦

Definice 6.1.4: Diskriminant polynomu p je vyjádřen předpisem

$$\mathrm{disc}(p) = \frac{\mathrm{res}(p, p')}{(-1)^{\frac{1}{2}n(n-1)}\, p_n}\,.\blacklozenge$$

Samotnou projekci množiny polynomů

$$A \subset R[x_1, x_2, ..., x_r], \text{ kde } r \geq 2,$$

pak provedeme následovně. Určíme množiny , L, S_1, S_2, přičemž platí:

$M = \{\mathrm{red}^k(p);\ p \in A \wedge k \geq 0 \wedge \deg(\mathrm{red}^k(p)) \geq 1\}$,

$L = \{\mathrm{lc}(p);\ p \in M\}$,

$S_1 = \{\mathrm{psc}_k(p, p');\ p \in M \wedge \deg(p') > k \geq 0\}$,

$S_2 = \{\mathrm{psc}_k(p, q);\ p, q \in M \wedge \min(\deg(p), \deg(q)) > k \geq 0\}$.

Projekční množinu P množiny polynomů A pak představuje sjednocení

$$P = L \cup S_1 \cup S_2.$$

Ukažme si zjištění projekční množiny na konkrétním příkladu s křivkami z Příkladu 5.1.

Příklad 6.1.1: Určete projekční množinu množiny polynomů $A = \{x^2 + y^2 - 4;\ x - y^2 + 1\}$ eliminováním proměnné y.

Podle předchozích definic pišme

$$p(x, y) = y^2 + x^2 - 4,\ q(x, y) = -y^2 + x + 1.$$

Prvním krokem je určení množiny M, která obsahuje polynomy p a q a všechna jejich redukta až do prvního stupně.

Platí:

$$\mathrm{red}(p) = p - \mathrm{lt}(p) = x^2 - 4$$

$$\mathrm{red}(q) = q - \mathrm{lt}(q) = x + 1$$

Vzhledem k tomu, že stupně polynomů $\deg(\mathrm{red}(p)) < 1$ a $\deg(\mathrm{red}(q)) < 1$, nezařazujeme je do množiny M, proto platí

$$M = A = \{y^2 + x^2 - 4; -y^2 + x + 1\}.$$

Při znalosti množiny M můžeme rovnou zkonstruovat množinu L, sestávající z vedoucích koeficientů polynomů množiny M. Těmi jsou $\mathrm{lc}(p) = 1$ a $\mathrm{lc}(q) = -1$, a tak množina

$$L = \{1, -1\}.$$

Pro určení množin S_1 a S_2 obsahujících základní subrezultanty již musíme provést rozsáhlejší přípravu, především určit Sylvestrovy matice pro příslušné polynomy vycházející z množiny M a poté vypočíst jejich determinanty.

Pro polynom $p = y^2 + x^2 - 4$ a jeho derivaci $p' = 2y$ pak existuje Sylvestrova matice ve tvaru

$$S_{p,p'} = \begin{pmatrix} 1 & 0 & x^2 - 4 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

a základní subrezultant

$$\mathrm{psc}_0(p, p') = \det(S_{p,p'}) = 4x^2 - 16.$$

Dále pro mnohočlen $q = -y^2 + x + 1$ a jeho derivaci $q' = -2y$ dostaneme Sylvestrovu matici

$$S_{q,q'} = \begin{pmatrix} -1 & 0 & x+1 \\ -2 & 0 & 0 \\ 0 & -2 & 0 \end{pmatrix},$$

jejíž determinant je roven

$$\mathrm{psc}_0(q, q') = \det(S_{q,q'}) = 4x + 4.$$

Tyto dva výsledky jsou prvky množiny

$$S_1 = \{4x^2 - 16; 4x + 4\}.$$

Nakonec ještě uveďme situaci pro polynomy p a q, pro něž má Sylvestrova matice tvar

$$S_{p,q} = \begin{pmatrix} 1 & 0 & x^2 - 4 & 0 \\ 0 & 1 & 0 & x^2 - 4 \\ -1 & 0 & x+1 & 0 \\ 0 & -1 & 0 & x+1 \end{pmatrix}$$

a základní subrezultant

$$\mathrm{psc}_0(p, q) = \det(S_{p,q}) = x^4 + 2x^3 - 5x^2 - 6x + 9.$$

Oproti předchozím dvěma případům, musíme nyní spočítat i subrezultant $\mathrm{psc}_1(p, q)$, který určíme jako determinant matice

$$S^1_{p,q} = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix},$$

která vznikla z matice $S_{p,q}$ škrtnutím dvou jejích posledních sloupců a druhého a posledního řádku. Potom tedy platí

$$\mathrm{psc}_1(p, q) = \det(S^1_{p,q}) = 0.$$

Celkově tak můžeme psát, že množina

$$S_2 = \{0; x^4 + 2x^3 - 5x^2 - 6x + 9\}$$

a konečně projekční množina

$$P = L \cup S_1 \cup S_2 = \{0; -1; 1; 4x + 4; 4x^2 - 16; x^4 + 2x^3 - 5x^2 - 6x + 9\}. \blacklozenge$$

Tento postup získání projekční množiny je označován jako Collinsova projekce, případně jeho výstup jako Collinsův operátor. Jednodušeji lze situaci popsat takto:

$$\mathrm{ProjC}_1 = \bigcup_{\substack{p\in A \\ q\in \mathrm{RED}(p)}} (\{\mathrm{lc}(q) \cup \mathrm{PSC}(q, q')), \quad \mathrm{ProjC}_2 = \bigcup_{\substack{p,q\in A \\ p<q}} \bigcup_{\substack{r\in \mathrm{RED}(p) \\ s\in \mathrm{RED}(q)}} \mathrm{PSC}(r, s),$$

kde $\mathrm{PSC}(p, q) = \{\mathrm{psc}_k(p, q);\ \min(\deg(p), \deg(q)) > k \geq 0,\ \mathrm{psc}_k(p, q) \neq 0\}$ a $\mathrm{RED}(p) = \{\mathrm{red}^k(p);\ \deg(p) \geq k \geq 0,\ \mathrm{red}(p) \neq 0\}$. Celá hledaná projekční množina je pak sjednocením

$$\mathrm{ProjC} = \mathrm{ProjC}_1 \cup \mathrm{ProjC}_2.$$

Vzhledem k nadměrné velikosti takto vzniklé projekční množiny, která obsahuje řadu nadbytečných prvků, se častěji užívají upravené postupy.

Hongova projekce (Hongův operátor) je dána rovností

$$\mathrm{ProjH} = \mathrm{ProjC}_1 \cup \mathrm{ProjH}_2,$$

přičemž množina

$$\mathrm{ProjH}_2 = \bigcup_{\substack{p,q\in A \\ p<q}} \bigcup_{r\in \mathrm{RED}(p)} \mathrm{PSC}(r, q).$$

Existuje též McCallumova projekce (McCallumův operátor), která je zapsána následovně:

$$\mathrm{ProjMC}_1 = \bigcup_{p\in A} \{p_i\},$$

kde p_i jsou koeficienty jednotlivých polynomů p z množiny A,

$$\mathrm{ProjMC}_2 = \bigcup_{p,q\in A} \{\mathrm{res}(p, q)\},$$

kde $\mathrm{res}(p, q)$ je determinant Sylvestrovy matice sestavené z polynomů p a q,

$$\mathrm{ProjMC}_3 = \bigcup_{p\in A} \{\mathrm{disc}(p)\},$$

kde $\mathrm{disc}(p)$ je podle výše popsané definoce diskriminant polynomu p, a konečně

$$\mathrm{ProjMC} = \mathrm{ProjMC}_1 \cup \mathrm{ProjMC}_2 \cup \mathrm{ProjMC}_3.$$

Pro výše uvedené projekce je pak možné vyjádřit i počty jejich prvků, které prokážou vyšší efektivnost upravených postupů. Pokud označíme počet polynomů množiny A jako m a n je maximum stupně polynomů z A, pak Collinsova projekce produkuje projekční množinu o maximálně m^2n^3 prvcích, zatímco Hongova projekční množina obsahuje maximálně m^2n^2 prvků a McCallumova dokonce méně než $m^2 + mn$ prvků.

Navíc u McCallumova operátoru lze provést další zlepšení, když množinu ProjMC_1 nahradíme množinou

$$\mathrm{ProjMC}_1' = \bigcup_{p \in A} \{\mathrm{lc}(p)\}.$$

V takovém případě je pak velikost projekční množiny menší než $m^2 + m$ prvků.

Podívejme se, jak McCallumova projekce funguje v praxi, použijeme k tomu již dříve použité polynomy

$$p = y^2 + x^2 - 4 \text{ a } q = -y^2 + x + 1.$$

Příklad 6.1.2: Nalezněte projekční množinu množiny $A = \{ y^2 + x^2 - 4; -y^2 + x + 1\}$ pomocí McCallumova operátoru.

Jednotlivé složky projekční množiny postupně budou:

$$\mathrm{ProjMC}_1' = \{\},$$

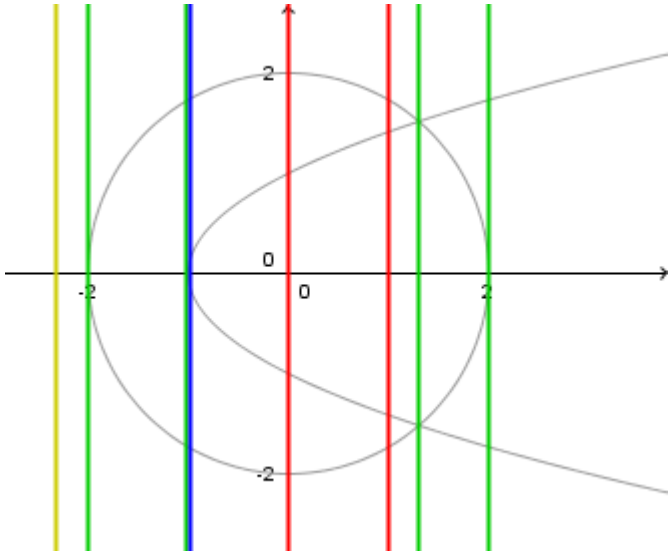
$$\mathrm{ProjMC}_2 = \{x^4 + 2x^3 - 5x^2 - 6x + 9\},$$

$$\mathrm{ProjMC}_3 = \{-4x + 16; 4x + 4\}.$$

Sjednocením dostaneme celou projekční množinu

$$\mathrm{ProjMC} = \{-4x + 16; 4x + 4; x^4 + 2x^3 - 5x^2 - 6x + 9\}. \blacklozenge$$

Jak lze vidět, oproti Collinsově metodě je výsledná projekční množina menší o prvky 0, –1 a 1, které byly buď nadbytečné anebo duplicitní s některým z dalších prvků. Pro úplnost ještě uveďme grafické znázornění situace.



Obrázek 12: Znázornění rozkladu podle projekční množiny

Pro lepší přehlednost jsou jednotlivé faktory Cillinsova a McCallumova operátoru barevně odlišeny. Zeleně jsou znázorněny ty složky projekčních množin, které se nakonec skutečně použijí pro rozklad, modrý je duplicitní faktor –1, červené složky 0 a

1 byly McCallumovou projekcí eliminovány a žlutě je označen faktor vycházející z členu

$$x^4 + 2x^3 - 5x^2 - 6x + 9$$

množiny ProjMC, který je nadbytečný, ale nepodařilo se jej odstranit. [ACM84a, DAV09, WIN96]

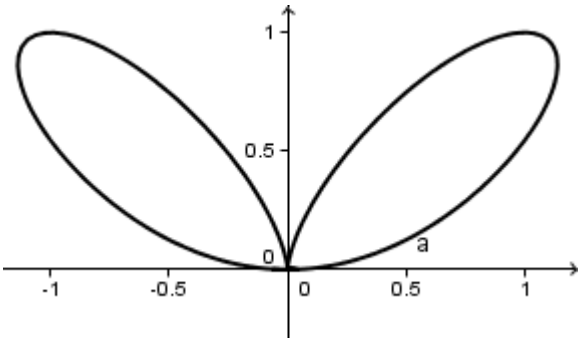
6.2 Konstrukce hald

Jak již bylo řečeno dříve, v této fázi cylindrické algebraické dekompozice je využito konstrukce a procházení stromových struktur vzniklých nad prostorem R^n.

Ukažme si, jak takový strom vypadá, na křivce s předpisem

$$a\text{: } x^4 - 2x^2y + y^4 = 0,$$

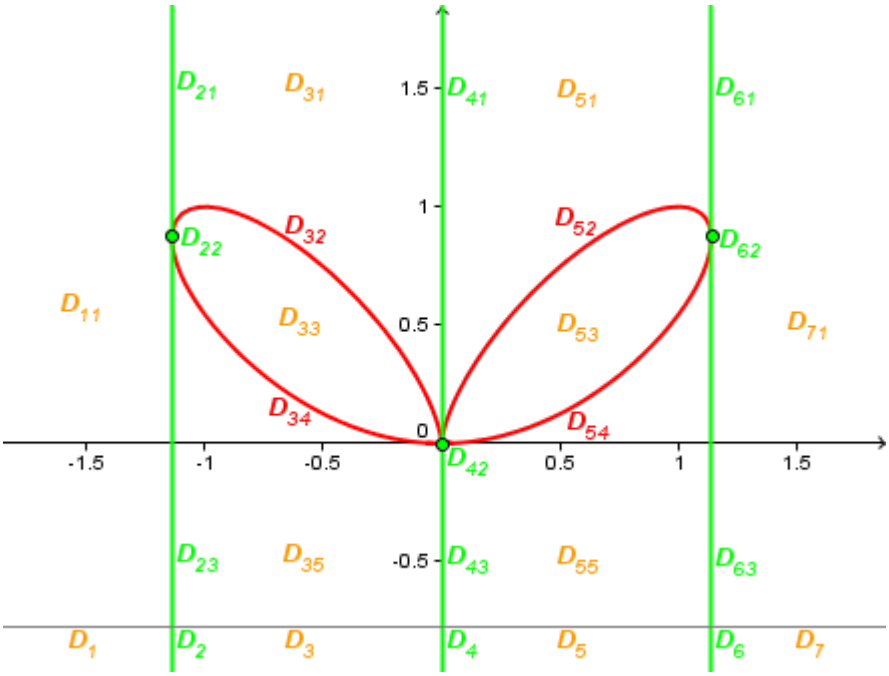
jíž odpovídá znázorněná křivka.



Obrázek 13: Grafické znázornění křivky $x^4 - 2x^2y + y^4 = 0$

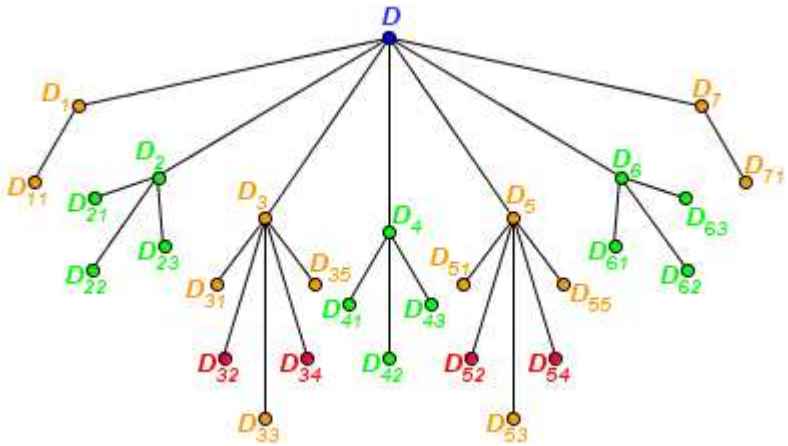
Provádíme-li rozdělení prostoru R^2, v němž s křivkou a pracujeme, cylindrickou algebraickou dekompozicí, obdržíme rozklad D sestávající z buněk

$$D_{11}, D_{21}, D_{22}, ..., D_{63} \text{ a } D_{71}.$$



Obrázek 14: Rozklad D prostoru R^2

Zakreslením příslušné stromové struktury můžeme tento rozklad popsat jiným způsobem. Kořenem takto vzniklého stromu je celý rozklad *D* (můžeme také psát R^2 nebo obecně R^n, což označuje prostor, ve kterém rozklad provádíme), který je rodičem pro další uzly stromu označující jednotlivé buňky rozkladu. Listy stromu jsou nakonec jednotlivé buňky kompletně provedené cylindrické algebraické dekompozice. V našem případě vypadá struktura stromu takto.



Obrázek 15: Stromová struktura rozkladu D prostoru R^2

Jakmile je strom pro CAD vytvořen, počítač jej projde a na jeho základě vytvoří množinu testovacích bodů jednotlivých buněk (testovací vzorek).

Všechny prvky této množiny jsou na počátku procházení prázdnými vektory, s průchodem přes každou hranu stromu směrem od kořene k listům je do něj v každém uzlu přidána jedna souřadnice. Po skončení procházení tak máme kompletní popis

celého prostoru R^n pomocí testovacích bodů, jejichž prostřednictvím se dá rozhodnout o pravdivostní hodnotě vstupní formule v dané buňce rozkladu.

Příklad 6.2.1: Proveďte konstrukci hald pro množinu polynomů $A = \{-x^3 + 2y^3 + 3y^2 - 3xy;\ x^2 + y^2 + 10x + 12y + 22\}$.

Zadanou množinu polynomů

$$\{-x^3 + 2y^3 + 3y^2 - 3xy;\ x^2 + y^2 + 10x + 12y + 22\}$$

lze popsat složkami projekční množiny

$\text{ProjMC}_1' = \{\}$,

$\text{ProjMC}_2 = \{31x^6 - 30x^5 + 24x^4 + 8144x^3 + 27978x^2 + 36168x + 20812\}$,

$\text{ProjMC}_3 = \{-108x^6 + 324x^4 + 324x^3 + 81x^2;\ 12x^2 + 96x + 120\}$,

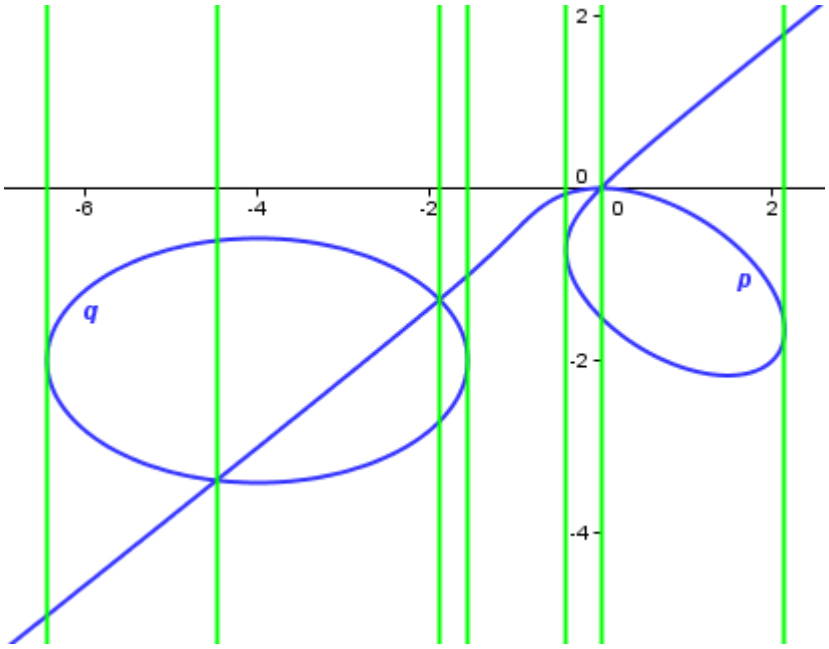
které produkují příslušné sekce rozkladu D

ProjMC_1': $x_1 \in \{\}$,

ProjMC_2: $x_2 \in \{-4{,}46041; -1{,}8762\}$,

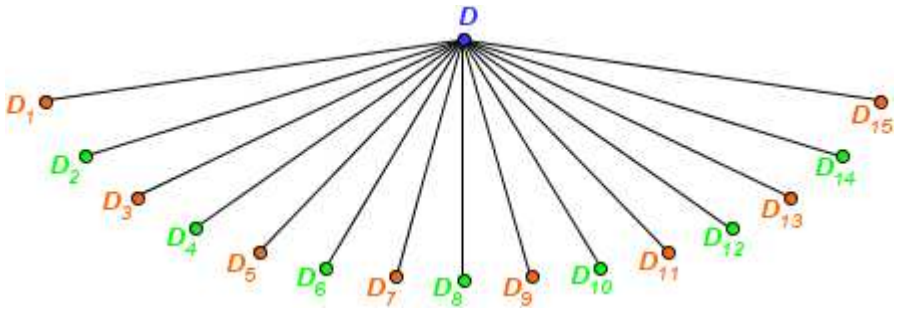
ProjMC_3: $x_3 \in \{-4-\sqrt{6}\ ; -4+\sqrt{6}\ ; -0{,}405204; 0; 2{,}13726\}$,

jak je vidět na obrázku.



Obrázek 16: Zvýrazněné sekce rozkladu D

Takto získané hodnoty využijeme při konstrukci stromu, ten bude zatím sestávat pouze z kořene a jeho přímých potomků, výška stromu je tedy 1.



Obrázek 17: Konstrukce 2. úrovně stromu

Nyní můžeme započít s „naplňováním“ množiny testovacích bodů. Na první úrovni kořene stromu existuje pouze prázdný vektor

$$\vec{v} = [\ ].$$

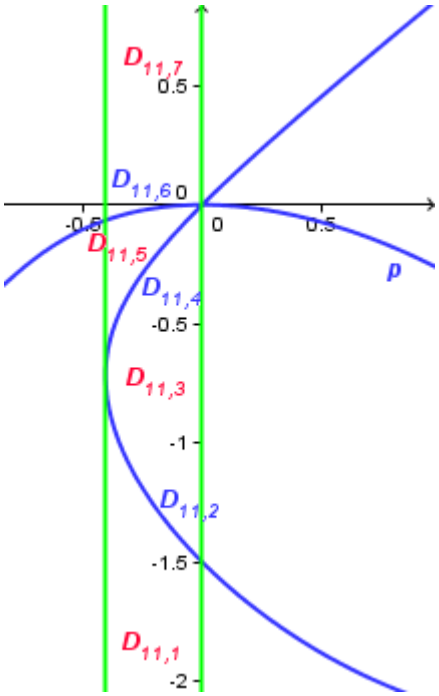
S přechodem do druhé úrovně vznikne 15 nových vektorů majících jednu souřadnici (nacházíme se v jednorozměrném prostoru). Přitom platí, že hodnota souřadnice těch vektorů, které přísluší sekcím rozkladu, je rovna hodnotě dané sekce:

$$\vec{v}_2 = [-4-\sqrt{6}\ ],\ \vec{v}_4 = [-4{,}46041],\ \vec{v}_6 = [-1{,}8762],\ ...,\ \vec{v}_{14} = [2{,}13726];$$

u vektorů náležejících sektorům je za souřadnici vybrána vhodná hodnota z vnitřku intervalu odpovídajícího sektoru, například:

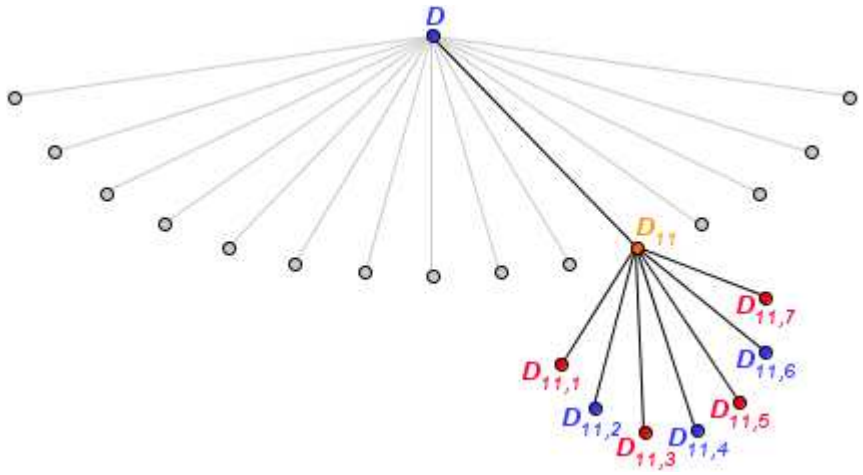
$$\vec{v}_1 = [-10],\ \vec{v}_3 = [-5],\ \vec{v}_5 = [-3],\ ...,\ \vec{v}_{13} = [1],\ \vec{v}_{15} = [3].$$

Dále je strom rozšířen o další úroveň. Vzhledem k tomu, že jsme ve dvojrozměrném prostoru R^2, budou uzly z této úrovně listy celého stromu. Takto vzniknou nové vektory, které z vektorů $\vec{v}_1, ..., \vec{v}_{15}$ zdědí první souřadnici, druhá bude odpovídat oblasti, kterou na dané sekci či v daném sektoru vytyčí jednotlivé křivky. Proveďme zmíněný krok na sektoru D_{11}, jenž je křivkou p rozdělen na 7 disjunktních oblastí.



Obrázek 18: Sektor D_{11}

Třetí úroveň stromu pro tuto oblast bude vypadat následovně.



Obrázek 19: Strom rozkladu se zvýrazněnou nižší úrovní sektoru D_{11}

Konstruujme vektory se dvěma souřadnicemi příslušné buňkám $D_{11,1}$, ..., $D_{11,7}$. Vektor $\vec{v}_{11,1}$ zastupuje oblast $D_{11,1}$, jeho první souřadnice odpovídá souřadnici vektoru $\vec{v}_{11}$ (vezměme $\vec{v}_{11} = [-\frac{1}{3}]$), druhá souřadnice musí být volena tak, abychom zůstali uvnitř oblasti $D_{11,1}$, kupříkladu

$$\vec{v}_{11,1} = [-\frac{1}{3}; -2].$$

V oblasti $D_{11,2}$ se pohybujeme po části křivky polynomu p omezené zleva a zprava sekcemi D_{10} a D_{12}. Vektor $\vec{v}_{11,2}$ opět dědí první souřadnici vektoru $\vec{v}_{11}$, jeho druhá souřadnice již musí být dopočítána z předpisu křivky p. Proto platí

$$\vec{v}_{11,2} = [-\frac{1}{3}; -0{,}309411].$$

Obdobným způsobem dopočítáme i zbývající buňky, čímž dostaneme kompletní množinu vektorů – testovacích bodů pro sektor D_{11}.♦

Dosazením souřadnic takto získaných testovacích bodů do polynomiálních rovnic a nerovnic vstupní formule pak již dokážeme rozhodnout o pravdivosti tvrzení v jednotlivých buňkách rozkladu D.

Než postoupíme k poslední fázi CAD – konstrukci výstupní formule, poznamenejme ještě, že ve své podstatě existují dva odlišné přístupy k využití cylindrické algebraické dekompozice, jenž se odlišují v průběhu fáze konstrukce hald.

Klasická CAD nejprve provede kompletní rozklad celého prostoru R^n, ve fázi konstrukce hald sestaví testovací vzorek a následně pomocí něj v každé buňce stanoví její pravdivostní hodnotu. Poté ve fázi konstrukce výstupní formule vygeneruje výstupní formuli.

Částečná cylindrická algebraická dekompozice také začne rozkladem prostoru R^n, ale oproti klasické CAD již přímo při konstrukci hald ověřuje pravdivostní hodnotu každé nově vzniklé buňky. Tím se dá zkrátit čas nutný k vyhodnocení výsledků a může se přistoupit rovnou k další fázi konstrukce výstupní formule. Jde o vylepšení algoritmu, které může urychlit získání výsledku, avšak tato přednost této metody se nemusí projevit vždy, funguje totiž jen u některých tříd kvantifikovaných formulí.

Demonstrujme výše zmíněný rozdíl na konkrétním příkladu.

Příklad 6.2.2: Ukažte rozdíl mezi klasickou a částečnou CAD na kvantifikované formuli $(\forall x)(\exists y)$: $x^2 + (y-2)^2 \geq 1 \vee x + y \geq 1$.

Při použití klasické CAD program provede rozklad, určí body testovacího cylindrického algebraického vzorku a teprve při následném hromadném vyhodnocování pravdivostních hodnot jednotlivých buněk zjistí, že v buňce D_{33} jsou obě nerovnosti nepravdivé. Tím pádem tedy není pravdivá ani celá zadaná formule, tudíž ekvivalentní formulí bez kvantifikátorů bude například rovnost

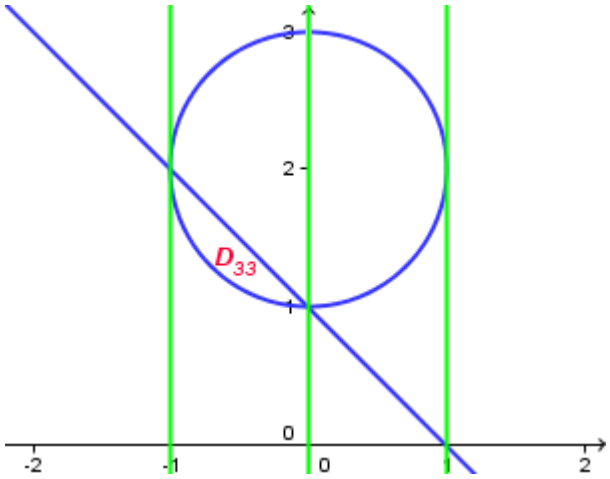
$$0 = 1$$

(počítačový program by takový výsledek vypsal jako výstup ve tvaru *False*).

Použijeme-li však částečnou CAD, provádí se zjištění pravdivosti dané buňky ihned po její konstrukci a získání souřadnic odpovídajícího testovacího bodu. Ve chvíli, kdy program zjistí, že v buňce D_{33} jsou obě nerovnosti nepravdivé, je zastaveno další konstruování hald i vyhodnocování pravdivosti, neboť je již zřejmé, že konjunkce zadaných nerovností nemůže být pravdivá pro všechna x, a následuje již rovnou poslední fáze celého algoritmu – konstrukce výstupní formule, jejímž výstupem je například opět rovnost

$$0 = 1$$

(výstup počítačového programu *False*).



Obrázek 20: Grafické znázornění křivek $x^2 + (y-2)^2 = 1$ a $x + y = 1$ ♦

Ukažme ještě úlohu reprezentující skupinu kvantifikovaných formulí, jejichž řešení nebude použitím částečné cylindrické algebraické dekompozice nikterak urychleno anebo pouze v zanedbatelné míře. Pro takovou ukázku nemusíme chodit daleko, stačí dokonce jen vhodným způsobem pozměnit předchozí formuli.

Příklad 6.2.3: Ukažte rozdíl mezi klasickou a částečnou CAD na kvantifikované formuli $(\forall x)\ x^2 + (y-2)^2 \geq 1 \vee x + y \geq -1$.

V případě klasické CAD proběhne celý proces tak, jak jsme již popsali. Program provede rozklad celého prostoru R^2 na jednotlivé buňky, poté je ve fázi konstrukce hald vytvořen testovací vzorek a následně je ověřena pravdivost tvrzení postupným dosazením jednotlivých testovacích bodů. Počítačový program odpoví výstupní nekvantifikovanou formulí ve tvaru

$$y \in Reals.$$

Tento výsledek můžeme interpretovat tak, že skutečně pro všechna x platí konjunkce uvedených nerovnic, přičemž druhá proměnná y nabývá reálných hodnot.

Na druhé straně částečná CAD započne rozkladem prostoru R^2, ve fázi konstrukce hald vždy po zkonstruování testovacího bodu příslušné buňky rozkladu okamžitě následuje jeho dosazení do formule a vyhodnocení pravdivosti. Avšak vzhledem k tomu, že proměnná x je kvantifikována obecným kvantifikátorem, je nutné, aby program vyhodnotil pravdivost při dosazení všech testovacích bodů cylindrického algebraického vzorku. Výstup bude taktéž ve tvaru

$$y \in Reals,$$

ale úspora času nutného pro provedení operace bude v tomto případě zanedbatelná.♦

[ACM84a, DAV09, WIN96]

6.3 Konstrukce výstupní formule

Poslední fází eliminace kvantifikátorů je konstrukce výstupní formule, přičemž se jedná o fázi výpočetně nejnáročnější. Přistoupit k ní můžeme více způsoby, z nichž každý má své klady i zápory. Podívejme se blíže na prostou metodu, Hongovu metodu a Collinsovu metodu.

Prostá metoda využívá ke konstruování výstupní formule složky projekční množiny, pro každou z nich definujme tzv. atomickou formuli.

Definice 6.3.1: Atomickou formulí rozumíme vyjádření ve tvaru

$$A(c, p) = \begin{cases} p < 0 \\ p = 0 \\ p > 0 \end{cases}$$

pro danou složku p projekční množiny

$$P = P_1 \cup P_2 \cup \ldots \cup P_k$$

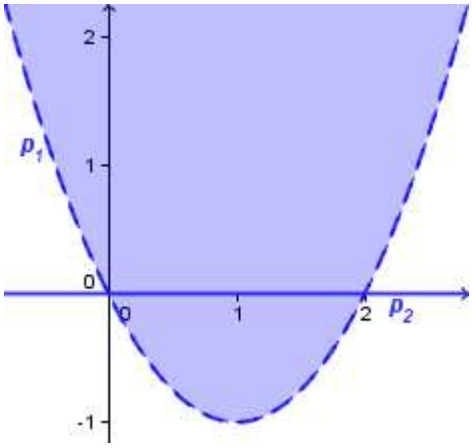
v příslušné buňce c rozkladu D.♦

Pomocí atomických formulí dokáže prostá metoda popsat každou buňku rozkladu D pravdivou vzhledem ke vstupní formuli F. Zároveň platí, že disjunkce konjunkcí těchto atomických formulí pro všechny složky projekční množiny a pro všechny buňky rozkladu D poskytuje projekční formuli, platí tedy

$$\mathrm{V}_{c \in D} \Lambda_{p \in P} A(c, p)\,.$$

Ta bývá, bohužel tomu tak ale není vždy, ekvivalentní s formulí vstupní. Pokud tomu tak je, nazývá se příslušná cylindrická algebraická dekompozice projekčně určenou.

Příklad 6.3.1: Nalezněte projekční formuli, jestliže víte, že projekční množina $P = \{p_1 = -x^2 + 2x + y;\quad p_2 = y\}$, přičemž vstupní formule je pravdivá ve zvýrazněných buňkách na obrázku.



Obrázek 21: Projekční množina se zvýrazněnými oblastmi, v nichž je vstupní formule pravdivá

Jednotlivé buňky, v nichž je formule pravdivá, popíšeme disjunkcí

$$[p_1 > 0 \wedge p_2 < 0] \vee [p_1 > 0 \wedge p_2 = 0] \vee [p_1 > 0 \wedge p_2 > 0],$$

která je projekční formulí. ♦

Vylepšením prosté metody je metoda Hongova. Z toho plyne i její nevýhoda, která spočívá v tom, že daná cylindrická algebraická dekompozice musí být projekčně určená. Je-li tomu tak, poskytuje Hongova metoda výstupní formuli v jednodušším tvaru nežli metoda Collinsova. Pro její použití je nutné zavést tzv. literály a funkce přechodu.

Definice 6.3.2: Mějme množinu

$$M = \{-; 0; +\}$$

a její podmnožinu S. Pravdivostní funkci l^S, pro kterou platí

$$l^S(x) = true$$

pro $x \in S$ a

$$l^S(x) = false$$

pro $x \notin S$, se nazývá literál. ♦

Definice 6.3.3: Definujme bijektivní zobrazení f z množiny formulí do množiny literálů, která vzorům

$$0 \neq 0,\ p < 0,\ p \leq 0,\ p = 0,\ p \geq 0,\ p > 0,\ p \neq 0,\ 0 = 0$$

přiřazuje po řadě obrazy

$$l^{\{\}},\ l^{\{-\}},\ l^{\{-,\,0\}},\ l^{\{0\}},\ l^{\{0,\,+\}},\ l^{\{+\}},\ l^{\{-,\,+\}},\ l^{\{-,\,0,\,+\}}.$$

Takovéto zobrazení se nazývá funkce přechodu. ♦

Pro funkci přechodu platí následující pravidla, kterých se využívá při aplikaci Hongovy metody konstrukce výstupní formule.

sčítání literálů: $l^S + l^T = l^{S \cup T}$

součin literálů: $l^S \cdot l^T = l^{S \cap T}$

Navíc platí rovnosti

$$l^{\{-,\,0,\,+\}} = true \text{ a } l^{\{\}} = false$$

a zároveň operace násobení je distributivní vůči operaci sčítání. Výstupní formule získaná Hongovou metodou je v minimalizovaném tvaru.

Pomocí funkce přechodu f je tedy možné vyjádřit složitější formuli literály, se kterými lze pak na základě výše zmíněných pravidel pracovat, v mnoha případech celý zápis zjednodušit a díky bijekci funkce f opět převést na tentokrát již jednodušší výstupní formuli.

Demonstrujme postup na následujícím úloze:

Příklad 6.3.2: Nalezněte Hongovou metodou výstupní formuli pro situaci z předchozího příkladu.

Aplikujme přechodovou funkci f na projekční formuli

$$[p_1 > 0 \wedge p_2 < 0] \vee [p_1 > 0 \wedge p_2 = 0] \vee [p_1 > 0 \wedge p_2 > 0],$$

dostaneme zápis

$$L = l_1^{\{+\}} \cdot l_2^{\{-\}} + l_1^{\{+\}} \cdot l_2^{\{0\}} + l_1^{\{+\}} \cdot l_2^{\{+\}}.$$

Tento součet součinů literálů se pokusme minimalizovat. Můžeme vytknout literál $l_1^{\{+\}}$ a máme

$$L = l_1^{\{+\}} \cdot (l_2^{\{-\}} + l_2^{\{0\}} + l_2^{\{+\}}) = l_1^{\{+\}} \cdot l_2^{\{-,\,0,\,+\}} = l_1^{\{+\}}.$$

Inverzní funkcí f^{-1} převedeme zápis na minimalizovanou výstupní formuli

$$p_1 > 0. \blacklozenge$$

Collinsova metoda oproti prosté a Hongově metodě poskytuje vždy výstupní formuli, která je ekvivalentní s formulí vstupní. Toho je docíleno tím, že v projekční fázi je určena tzv. rozšířená projekční množina, která obsahuje i derivace polynomů z množiny A. Tímto způsobem stanovený rozklad D je pak již vždy projekčně určen. Nevýhodou této metody je však vyšší výpočetní náročnost, jelikož narůstá množství polynomů, se kterými je třeba pracovat, i počet prováděných operací. [DAV09, WIN96]

6.4 Formule s více proměnnými

Zatím jsme se zabývali pouze matematickými formulemi s jednou či dvěma proměnnými. Jejich dokazování, či obecně řešení matematických problémů, které lze právě na eliminaci kvantifikátorů převést, je poměrně snadné. Zároveň jednorozměrné a dvojrozměrné křivky vystupující v těchto formulích jsou dosti snadno graficky znázornitelné, což posloužilo při ilustraci jednotlivých doprovodných příkladů a ukázalo, jakým způsobem je celá CAD prováděna a jak je získán vzorek testovacích bodů.

Pomocí cylindrické algebraické dekompozice však jde provádět eliminaci kvantifikátorů i pro formule s více proměnnými, jinými slovy eliminaci můžeme aplikovat též ve vícerozměrných prostorech, které již není možné graficky zobrazit, případně je zobrazíme jen s obtížemi. Podívejme se na zápis obecného algoritmu CAD, který provedení dekompozice v těchto prostorech umožňuje. Vstupními daty algoritmu je množina zadaných polynomů (resp. nenulové strany rovnic zadané homogenní soustavy rovnic) $A \subset R[x_1, x_2, ..., x_r]$ a číslo r značící počet proměnných v polynomech, výstupem je cylindrický algebraický vzorek s obsahující jednotlivé testovací body a soubor atomických formulí F. [WIN96]

6.5 Obecný algoritmus CAD

Obecný algoritmus cylindrické algebraické dekompozice se skládá ze dvou hlavních částí, v nichž jsou použity matematické postupy a metody popsané v kapitolách 5.1 až 5.3. Která z těchto částí algoritmu bude provedena, je určeno na základě počtu rozměrů daného prostoru R^r, v němž je dekompozice prováděna. Pokud je $r = 1$, provede se část (1) algoritmu, je-li $r > 1$, započne část (2), v níž je pak algoritmus volán rekurzivně.

Algoritmus CAD (vstupní data algoritmu: množina polynomů $A \subset R[x_1, x_2, ..., x_r]$, dimenze daného prostoru r; výstupní data algoritmu: cylindrický algebraický vzorek s, soubor atomických formulí F)**:**

begin

s je prázdná množina;

(1) if $r = 1$:

vypočti reálné kořeny polynomů z množiny A;

urči cylindrický algebraický vzorek s pro A-invariantní cylindrickou algebraickou dekompozici D;

zkonstruuj soubor atomických formulí F;

return s, F;

(2) if $r > 1$:

urči projekční množinu Proj(A) vstupní množiny A;

dekrementuj r;

zavolej rekurzivně algoritmus CAD se vstupními daty Proj(A) (výstupními daty tohoto rekurzivního volání bude cylindrický algebraický vzorek s');

nechť $s' = (s'_1, s'_2, ..., s'_t)$, přičemž $s'_j = (s'_{j,1}, s'_{j,1}, ..., s'_{j,r-1})$, kde $1 \leq j \leq t$;

urči cylindrický algebraický vzorek s pro A-invariantní cylindrickou algebraickou dekompozici D určením reálných kořenů všech polynomů $p(s'_{j,1}, s'_{j,1}, ..., s'_{j,r-1}, x_r) \in A$ v proměnné x_r;

rozšiř stávající soubor atomických formulí F;

return s, F;

end.

Takto uvedený algoritmus CAD přirozeně nedokáže provést eliminaci kvantifikátorů jako takovou. Je potřeba jej zakomponovat do komplexnějšího algoritmu, který ze zadané kvantifikované formule φ vyčlení její nekvantifikovanou část φ^*, která je ve své podstatě již soustavou rovnic a nerovnic A, se kterou pracuje algoritmus CAD. Výše uvedené vztahy můžeme schématicky zapsat takto

$$\varphi = (Q_{k+1}x_{k+1})...(Q_r x_r)\varphi^*(x_1, x_2, ..., x_r),$$

kde Q_i, $k + 1 \leq i \leq r$, je jeden z kvantifikátorů $\forall$ anebo $\exists$, a $A = \varphi^*$.

Výstupy algoritmu CAD po jeho provedení jsou cylindrický algebraický vzorek s a soubor atomických formulí F, jež v závěrečné fázi algoritmu eliminace kvantifikátorů poslouží ke zkonstruování výstupní nekvantifikované formule ψ. Ta je již ekvivalentní se zadanou formulí φ.

Algoritmus eliminace kvantifikátorů (vstupní data algoritmu: formule s kvantifikátory φ; výstupní data algoritmu: ekvivalentní formule bez kvantifikátorů ψ)**:**

begin

(1) ze vstupní kvantifikované formule φ extrahuj nekvantifikovanou formuli φ^*, pro kterou platí $A = \varphi^*$;

(2) zavolej **algoritmus CAD** se vstupními daty A, nechť jeho výstup je s a F;

(3) zkonstruuj nekvantifikovanou formuli ψ za použití s a F;

(4) return ψ;

end.

Pro úplnost ještě uveďme vývojové diagramy algoritmu CAD i celé eliminace kvantifikátorů za použití cylindrické algebraické dekompozice, které jsou znázorněny v přílohách práce (Příloha 1, 2). [WIN96]

6.6 Vylepšený algoritmus korektní cylindrické algebraické dekompozice

Zajímavé vylepšení algoritmu cylindrické algebraické dekompozice spočívá v určení přilehlých buněk rozkladu. Neformálně můžeme přilehlé buňky definovat tak, že se dotýkají, neboli leží vedle sebe, formálně jsou takové buňky definovány prostřednictvím svého sjednocení, které musí tvořit jednu nerozdělenou oblast.

Právě využití přilehlosti buněk v CAD se věnovali George E. Collins s kolegy Dennisem Arnonem a Scottem McCallumem v článku *Cylindrical Algebraic Decomposition II: An Adjacency Algorithm for the Plane* vydaném v roce 1984 v *SIAM Journal on Computation*.

Z hlediska cylindrické algebraické dekompozice je informace o přilehlosti buněk poměrně důležitý údaj, protože může být použita k eliminování některých kroků původního algoritmu rozkladu a snížit tak jeho časovou náročnost.

Poznamenejme také, že tento vylepšený algoritmus se poněkud odlišuje od původního prezentovaného v předchozím článku Collinse a jeho kolegů.

Pro pořádek definujme pojem přilehlosti.

Definice 6.6.1: Jsou-li oblasti D_1 a D_2 rozkladu D přilehlé (tedy jejich sjednocení tvoří jednu nerozdělenou oblast), nazýváme množinu $\{D_1; D_2\}$ jako přilehlost. Jsou-li obě zmíněné oblasti sekce rozkladu, pak jde o přilehlost sekce-sekce. ♦

Zároveň lze takto definované přilehlosti rozdělit do dvou tříd podle haldy, ve které se nacházejí (resp. podle toho, zda se nachází ve stejné buňce „nadřazené“ nebo ve dvou „nadřazených“ buňkách sousedních). Pokud se obě buňky nalézají ve stejné haldě, pak se jedná o vnitrohaldovou přilehlost (intrastack adjacency), jsou-li každá z jiné haldy, mluvíme o mezihaldové přílehlosti (interstack adjacency).

Určení přilehlostí prvního typu je vcelku jednoduché. Stačí uvážit, že každá halda je rozdělena do konečného množství buněk – sektorů a sekcí, přičemž každý ze sektorů přiléhá k právě jedné sekci „nad“ ním a právě k jedné sekci „pod“ ním. Z toho plyne, se

přilehlosti uvnitř určité haldy mohou být pouze jednoho z těchto dvou druhů a na toto určení stačí libovolný algoritmus cylindrické algebraické dekompozice.

Zjišťování mezihladové přilehlosti je již podstatně složitější, proto se jí budeme blíže věnovat.

Nejprve definujme tzv. korektní cylindrickou algebraickou dekompozici.

Definice 6.6.2: Cylindrická algebraická dekompozice D prostoru R^r je korektní, pokud platí:

1) existuje takový polynom $p \in R[x_1, ..., x_r]$, jehož množina nulových bodů je ekvivalentní s množinou sekcí rozkladu D, a

2) indukovaný rozklad D' prostoru R^{r-1} je také korektní, pro $r > 1$. ♦

Pokud takový mnohočlen p existuje, nazýváme jej definující polynom.

Obecně pak platí, že pokud D je korektní CAD prostoru R^r, $r > 2$, tak i rozklad D' je korektní, zároveň jakýkoliv definující polynom $p \in R[x_1, ..., x_r]$ rozkladu D je diskretizovatelný[10] na každé buňce $c \in D'$ a konečně pro definující polynom p rozkladu D je $D = \bigcup_{c \in D'} S(p, c)$, kde $S(p, c)$ je halda nad buňkou c, která je podmnožinou rozkladu D. Navíc též platí i tvrzení obrácené, že je-li D' korektní rozklad, existuje definující polynom $p \in R[x_1, ..., x_r]$, který je diskretizovatelný na každé buňce $c \in D'$, a $D = \bigcup_{c \in D'} S(p, c)$, pak je i D korektní.

Dále je vhodné provést rozšíření prostoru R jeho sjednocením s dvouprvkovou množinou $\{-\infty; \infty\}$, přičemž píšeme $R^* = R \cup \{-\infty; \infty\}$. Rozšíření provedeme i pro všechny cylindry, které se v rozkladu objeví, přibudou tedy dvě sekce, nazvěme je $-\infty$-sekce a $+\infty$-sekce. Cylindr nad určitou oblastí D_i tentokrát budeme značit $C^*(D_i)$. Jako S^* označme rozšíření příslušné haldy S.

Pro označení nových sekcí použijme následující indexování: Je-li halda S označena indexem i a obsahuje j sekcí, pak buňky odpovídající $-\infty$-sekci a $+\infty$-sekci indexujeme jako $(i, 0)$ a $(i, 2j + 2)$.

Za jádro korektní cylindrické algebraické dekompozice lze považovat algoritmus, který nazveme algoritmus ADJ, který můžeme rozepsat následovně.

---

[10] Polynom p je diskretizovatelný na oblasti D_i prostoru R^{n-1}, jestliže část množiny jeho nulových bodů ležících v cylindru $C(D_i)$ představuje k disjunktních sekcí cylindru $C(D_i)$, $k \geq 0$.

algoritmus ADJ (vstupní data algoritmu: polynom $p \in R[x, y]$, reálné číslo α takové, že $p(\alpha, y) \neq 0$, racionální čísla b_1 a b_2 taková, že $b_1 < \alpha < b_2$, přičemž p je diskretizovatelný v oblasti $D_1 = \langle b_1, \alpha)$ a zároveň p je diskretizovatelný v oblasti $D_2 = (\alpha, b_2\rangle$, výstupní data algoritmu: seznam S_1 všech mezihaldových přilehlostí typu sekce-sekce mezi haldami $S^*(p, c^0)$ a $S^*(p, D_1)$, seznam S_2 všech mezihaldových přilehlostí typu sekce-sekce mezi haldami $S^*(p, c^0)$ a $S^*(p, D_2)$)**:**

begin

S_1 je prázdná množina;

S_2 je prázdná množina;

urči reálné kořeny $k_1, ..., k_m$ $(m \geq 0)$ polynomu $p(\alpha, y)$ a racionální hodnoty $h_0, ..., h_m$ tak, že $h_0 < k_1 < h_1 < ... < k_m < h_m$ (*pro $m = 0$ polož h_0 rovno libovolnému racionálnímu číslu*);

do proměnné u ulož hodnotu b_1;

do proměnné v ulož hodnotu b_2;

dokud existuje h_j, $0 \leq j \leq m$, takové, že $p(x, h_j)$ má v intervalu $\langle u, v\rangle$ reálný kořen, ulož do b^* racionální (přibližnou) hodnotu středu intervalu (u, v) různou od α;

přiřaď do u, v hodnoty b_1, b^* anebo b^*, b_2 tak, aby v intervalu (u, v) leželo α;

(8) do n ulož počet reálných kořenů $p(u, y)$ v intervalu $(-\infty; h_0)$;

ulož do seznamu S_1, že sekce 0 haldy $S^*(p, c^0)$ je přilehlá sekcím 0, 1, ..., n haldy $S^*(p, D_1)$;

pro j od 1 do m udělej:

do n_j ulož počet reálných kořenů $p(u, y)$ v intervalu $(h_{j-1}; h_j)$;

ulož do seznamu S_1, že sekce j haldy $S^*(p, c^0)$ je přilehlá sekcím $n + 1$, ..., $n + n_j$ haldy $S^*(p, D_1)$;

do n ulož $n + n_j$;

do n_{m+1} ulož počet reálných kořenů $p(u, y)$ v intervalu (h_m, ∞);

ulož do seznamu S_1, že sekce $m + 1$ haldy $S^*(p, c^0)$ je přilehlá sekcím $n + 1$, ..., $n + n_{m+1}$, $n + n_{m+1} + 1$ haldy $S^*(p, D_1)$;

vrať se na (8) a zopakuj uvedené kroky pro oblast D_2 místo D_1 a pro seznam S_2 místo S_1;

end.

Uveďme nyní celý vylepšený algoritmus pro korektní cylindrickou algebraickou dekompozici v rovině R^2.

Algoritmus korektní CAD (vstupní data algoritmu: množina polynomů $A \subset R[x, y]$, dimenze daného prostoru r; výstupní data algoritmu: seznam I indexů buněk korektního A-invariantního rozkladu D, seznam všech přilehlostí P rozkladu D, cylindrický algebraický vzorek s)**:**

begin

urči projekční množinu Proj(A) vstupní množiny A;

vypočti reálné kořeny ireducibilních faktorů nenulových prvků Proj(A) a urči 0-rozměrné buňky indukovaného rozkladu D';

urči testovací bod pro každou buňku rozkladu D';

přiřaď do A^* primitivní část součinu nenulových prvků A;

nechť $s_1 < s_2 < ... < s_{2n+1}$, $n \geq 0$, jsou testovací body rozkladu D' (s_{2i+1} *jsou racionální testovací body pro 1-rozměrné buňky,* s_{2i} *jsou testovací body pro 0-rozměrné buňky*);

I je prázdná množina;

P je prázdná množina;

pro i od 1 do $2n + 1$ udělej:

urči reálné kořeny $A^*(s_i, y)$, tím budou známy sekce haldy T v R^2;

urči indexy buněk v haldě T a přidej je do seznamu I;

přidej vnitrohaldové přilehlosti haldy T do seznamu L;

pro i od 1 do n udělej:

zavolej **algoritmus ADJ** se vstupními daty A^*, s_{2i}, s_{2i-1} a s_{2i+1};

jeho výstup S_1 a S_2 ulož do P (*zde je pravděpodobné, že bude nutné výstup transformovat tak, aby indexy odpovídaly příslušným buňkám*);

z obsahu seznamu P odvoď zbývající mezihaldové přilehlosti rozkladu D a ulož je do P;

s je prázdná množina;

použij testovací body rozkladu D' k určení bodů cylindrického algebraického vzorku a ulož je do s;

end.

Poznámka: Primitivní částí polynomu p v proměnné x rozumíme polynom $pp(f)$ vycházející ze vztahu $pp(f) = \frac{p}{\text{cont}(p)}$, kde cont($p$) je největší společný dělitel všech koeficientů polynomu p.♦

Obdobné vylepšení algoritmu cylindrické algebraické dekompozice lze provést i pro vícerozměrné prostory, zde se však již jedná o velmi komplikovanou záležitost. Bližší informace k výše zmíněnému vylepšení v rovině lze nalézt v [ACM84b].

7 Matematické programy pracující s CAD

Přestože výpočetní technika a matematické programy práci s eliminováním kvantifikátorů značně ulehčují, mohou se objevit problémy vycházející z velkého množství výpočtů potřebných pro její aplikaci, zvláště pak obsahuje-li kvantifikovaná formule více proměnných, případně jsou-li tyto proměnné ve formuli obsaženy ve vyšších mocninách.

Neocenitelným pomocníkem se v takovém případě jeví programy počítačové algebry, které jsou schopny eliminaci kvantifikátorů provádět a to i v poměrně složitých případech. V současné době mezi tyto programy patří Wolfram Mathematica společnosti Wolfram Research, Maple od společnosti Waterloo Maple a aplikace QEPCAD.

Zatímco Mathematica a Maple jsou komplexní matematická výpočetní prostředí, v nichž eliminace kvantifikátorů tvoří jen jednu z mnoha jejich funkcí, program QEPCAD je určen především pro práci s matematickými formulemi a jejich zjednodušování prostřednictvím odstraňování kvantifikátorů. Další rozdíl lze u výše jmenovaných programů nalézt i v jejich dostupnosti, Mathematica i Maple jsou programy komerční a jsou dostupné pro velkou škálu operačních systémů (jmenovitě MS Windows, Linux, Mac OS a Solaris), naproti tomu aplikace QEPCAD je program volně šiřitelný, ovšem spustitelný pouze v systémech Linux, Solaris a Mac OS, což do jisté míry ztěžuje jeho dostupnost uživatelům přivyklým práci v operačních systémech Windows.

V neposlední řadě je také nutné zmínit i přístup k provedení celé eliminace kvantifikátorů, v tomto směru je mezi programy významný rozdíl. Programy Mathematica a QEPCAD spoléhají na již starší, avšak stále vylepšované a optimalizované verze algoritmu cylindrické algebraické dekompozice, naproti tomu program Maple pracuje s trojúhelníkovou dekompozicí, nicméně má i příkazy pro cylindrickou algebraickou dekompozici.

Mluvíme-li o eliminaci kvantifikátorů, nemusí jít pouze o dokazování samotných kvantifikovaných formulí či matematických vět, tato problematika zahrnuje i další oblasti matematiky (kupříkladu řešení rovnic a nerovnic, jejich soustav, práce s limitami funkcí, vyšetřování průběhů funkcí, analytickou geometrii a podobně). Důležitou podmínkou při této práci je však přesnost a jasnost příkazů zadávaných matematickému programu.

Ukažme si na několika příkladech v jednotlivých programech, jak eliminace kvantifikátorů vypadá v praxi a jak ji můžeme využít. [BRO02, BRO03, MAP12, WOL12]

7.1 Wolfram Mathematica

7.1.1 Program Wolfram Mathematica

Začněme prací v programu Wolfram Mathematica.

Wolfram Mathematica je výpočetní software původně navržený britským vědcem Stephenem Wolframem a následně vyvíjený společností Wolfram Research se sídlem v Champaign v Illinois. První verze programu byla pro uživatele přístupná od roku 1988 a od té doby dospěla již do své osmé verze, přičemž v každé nové verzi přinesla nové nástroje a funkce. Pro nás zajímavá eliminace kvantifikátorů v matematických větách se objevila ve verzi Mathematica 5.[11] (WOL12)

7.1.2 Příkazy programu Wolfram Mathematica

Mezi nejdůležitější příkazy programu Wolfram Mathematica pro práci s kvantifikátory patří příkaz **ForAll [x, tvrzení]** (případně **ForAll [x, podmínka, tvrzení]**, pokud má pro zadanou proměnnou *x* platit jistá další podmínka, či **ForAll [{x1, x2, ...}, tvrzení]**, když pracujeme s uspořádanou *n*-ticí proměnných x_1, x_2, ...), jenž je užíván jako obecný kvantifikátor ∀ (velký kvantifikátor, „pro všechny prvky platí“), a příkaz **Exists [x, tvrzení]** (případně **Exists [x, podmínka, tvrzení]** nebo **Exists [{x1, x2, ...}, tvrzení]**) reprezentující existenční kvantifikátor ∃ (malý kvantifikátor, „existuje takový prvek, pro který platí“). Chceme-li například zadat matematickou formuli odpovídající zápisu $(\forall x)\,(\exists y)$: $x^2 - y^2 + xy - 5 \geq 0$, využijeme kombinaci právě těchto příkazů ve tvaru **ForAll [x, Exists [y, x^2 – y^2 + x*y – 5 >= 0]]**. Program na potvrzení přijatého příkazu vypíše na obrazovku zápis $\forall_x \exists_y$ `–5+x^2+xy–y^2≥0`.

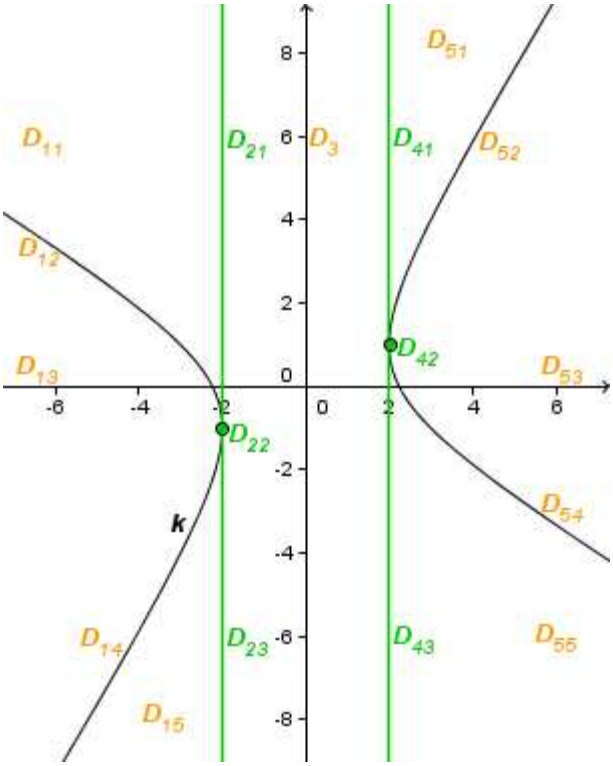
Chceme-li do zadávané formule zapsat soustavu rovnic či nerovnic, je pak nutné použít logické spojky AND pro logickou operaci konjunkci a OR pro disjunkci, přičemž spojka AND je v programu Mathematica reprezentována zápisem **&&** a spojku OR zastupuje zápis ||.

[11] Samotné jméno „Mathematica“ nepochází od Stephena Wolframa ani žádného z jeho kolegů, ale navrhl jej Steve Jobs, který s Wolframem udržoval přátelství.

Nejprve se zaměřme na samotný algoritmus cylindrické algebraické dekompozice, jehož prostřednictvím lze získat cylindrický algebraický vzorek pro zadané formule. Můžeme použít příkaz **CylindricalDecomposition [nerovnice, {x1, x2, ...}]**, který vypíše rozklad celého prostoru do jednotlivých disjunktních buněk. Tato jednoduchá operace však může mít výstup, jenž bude relativně složité interpretovat, navíc výsledek zahrnuje jen buňky, které splňují nerovnici. Ukažme to v návaznosti na předchozí úlohu na polynomu $x^2 - y^2 + xy - 5$. Samotný rozklad je nutné provést nejlépe ve třech oddělených krocích, protože argumentem příkazu **CylindricalDecomposition** smí být pouze nerovnice, případně rovnice, nikoliv samotný polynom. Proto píšeme zadání nejprve ve tvaru **CylindricalDecomposition [x^2 – y^2 + x*y – 5 > 0, {x, y}]**, dostaneme výstup (x<–2&&$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$<y<$\frac{x}{2}+\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$)||(x>2 &&$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$<y<$\frac{x}{2}+\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$), dále pak **CylindricalDecomposition [x^2 – y^2 + x*y – 5 == 0, {x, y}]**, obdržíme rozklad ve tvaru (x<–2&&(y==$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$||y==$\frac{x}{2}+\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$))|| (x==–2&&y==–1)||(x==2&&y==1)||(x>2&&(y==$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$||y== $\frac{x}{2}+\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$)) a aby byl výčet všech buněk kompletní, je do třetice nutné zadat ještě příkaz **CylindricalDecomposition [x^2 – y^2 + x*y – 5 < 0, {x, y}]**, na nějž program odpoví řetězcem (x≤–2&& (y<$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$||y>$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$))||–2<x<2||(x≥2&&(y<$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$||y>$\frac{x}{2}-\frac{1}{2}\sqrt{5}\sqrt{-4+x^2}$)).

Interpretace výsledků se může zdát komplikovaná, nicméně není tomu tak. První výstup určuje dvě buňky, z nichž pro jednu platí $x < -2$ a y je shora a zdola omezeno hodnotami $\frac{x}{2} \pm \frac{1}{2}\sqrt{5} \cdot \sqrt{-4+x^2}$ (označme ji D_{13}) a pro druhou $x > 2$, přičemž pro y opět platí omezení shora i zdola hodnotami $\frac{x}{2} \pm \frac{1}{2}\sqrt{5} \cdot \sqrt{-4+x^2}$ (značíme D_{53}). Druhý výstup popisuje více buněk, pro $x < -2$ je hodnota y přímo rovna $\frac{x}{2} \pm \frac{1}{2}\sqrt{5} \cdot \sqrt{-4+x^2}$

(popisujeme tedy dvě oddělené části jedné větve zadané hyperboly; označme je D_{12} a D_{14}), dále dva samostatné body o souřadnicích $x = -2$, $= -1$ (značme jej D_{22}) a $x = 2$, $y = 1$ (buňka D_{42}) a konečně pro $x > 2$ opět dvě oddělené části druhé větve hyperboly, kdy $y = \frac{x}{2} \pm \frac{1}{2}\sqrt{5} \cdot \sqrt{-4 + x^2}$ (označíme D_{52} a D_{54}). Poslední, třetí výstup pak popisuje zbývající buňky, které jsou pro hodnoty $x \leq -2$ a $x \geq 2$ dokonce sloučené, neboť po řadě sdružují případy $x < -2$, $x = -2$ a $x > 2$, $x = 2$. Tyto buňky reprezentují ty oblasti, které pro y splňují buď $y > \frac{x}{2} + \frac{1}{2}\sqrt{5} \cdot \sqrt{-4 + x^2}$ anebo $y < \frac{x}{2} - \frac{1}{2}\sqrt{5} \cdot \sqrt{-4 + x^2}$ (značme je popořadě D_{11}, D_{15}, D_{21}, D_{25}, D_{41}, D_{45}, D_{51} a D_{55}). Mimoto dostáváme vyjádřením řetězce nerovností $-2 < x < 2$ poslední buňku D_3. Situace pak vypadá následovně.



Obrázek 22: Dekompozice R^2 podle křivky určené rovnicí $x^2 - y^2 + xy - 5 = 0$

Po takto provedeném rozkladu by v algoritmu cylindrické algebraické dekompozice následovalo naplnění seznamu testovacího bodů, čili fáze konstrukce hald prostřednictvím konstruování a procházení stromu vzniklého nad rozkladem D. Pro vygenerování seznamu testovacího vzorku existuje příkaz **SemialgebraicComponentInstances [nerovnice, {x1, x2, ...}]**, který provede rozklad a vrátí alespoň jeden testovací bod pro každou oblast zadané nerovnice.

Nevýhodou je ale jako v předchozím postupu opět fakt, že navrácené testovací body odpovídají pouze oblastem, v nichž je zadaná nerovnice pravdivá. Je tedy nutné postupovat po částech a uplatnit příkaz postupně, nejlépe po řadě pro $x^2 - y^2 + xy - 5 > 0$, $x^2 - y^2 + xy - 5 = 0$ a nakonec pro $x^2 - y^2 + xy - 5 < 0$.

Při práci s nerovnicí $x^2 - y^2 + xy - 5 > 0$ píšeme **SemialgebraicComponentInstances [x^2 – y^2 + x*y – 5 > 0, {x, y}]** a program odpoví výpisem množiny testovacích bodů `{{x→–3,y→0},{x→3,y→0}}`. Máme tedy dva body testovacího vzorku s = ([–3; 0], [3; 0]).

Dáme-li nově získané informace dohromady s dříve získaným rozkladem D podle polynomu $x^2 - y^2 + xy - 5$, zjistíme, že testovací bod [–3; 0] zastupuje celou buňku D_{13}, zatímco bod [3; 0] reprezentuje buňku D_{53}.

Pro úplnost můžeme celou situaci zobrazit v grafu i se zvýrazněnými testovacími body pomocí příkazu **Show [RegionPlot [x^2 – y^2 + x*y – 5 > 0, {x, –10, 10}, {y, –10, 10}], ListPlot [{x, y}/.%]]**, kde **RegionPlot [nerovnice, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}]** vykreslí samotný graf nerovnice a příkazem **ListPlot [{x1, x2, ...}]** jsou do něj zaneseny zjištěné testovací body.

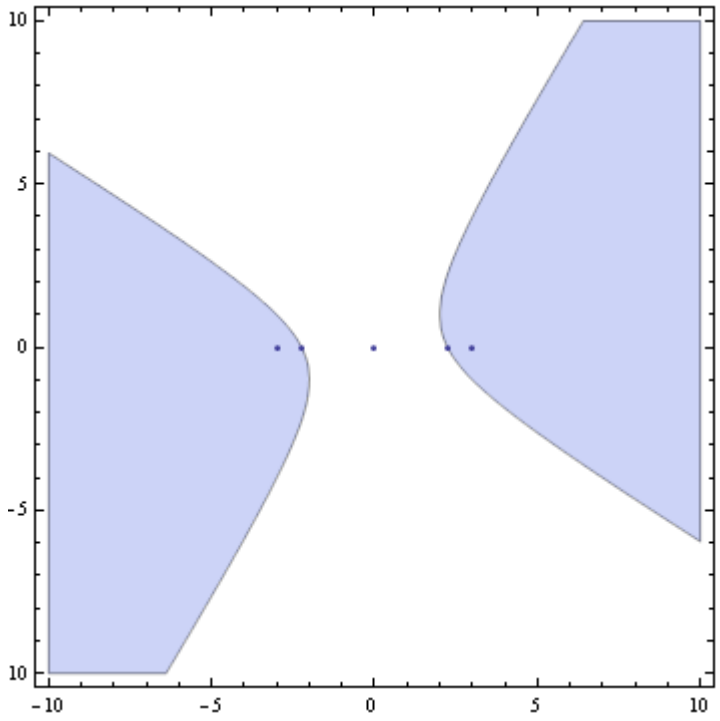
Jak již bylo řečeno, výše popsaná situace ale odpovídá zadané nerovnici, tedy $x^2 - y^2 + xy - 5 > 0$, a pro tuto část roviny jsou vypočítány i testovací body. Chceme-li získat i zbývající body pro část roviny, která nerovnici neodpovídá, je třeba použít pozměněný příkaz **SemialgebraicComponentInstances [x^2 – y^2 + x*y – 5 == 0, {x, y}]** a dále pak ještě **SemialgebraicComponentInstances [x^2 – y^2 + x*y – 5 < 0, {x, y}]**, čímž pokryjeme i zbývající část roviny a nalezneme testovací body i pro ostatní buňky rozkladu D.

Pro první z příkazů **SemialgebraicComponentInstances [x^2 – y^2 + x*y – 5 == 0, {x, y}]** dostáváme `{{x→–√5,y→0},{x→√5,y→0}}` a testovací vzorek s tak doplníme o body $[-\sqrt{5}; 0]$ a $[\sqrt{5}; 0]$. Pro poslední příkaz ve tvaru **SemialgebraicComponentInstances [x^2 – y^2 + x*y – 5 < 0, {x, y}]** je odpovědí programu výpis `{{x→–3,y→–7},{x→–3,y→3},{x→0,y→0},{x→3,y→-3},{x→3,y→7}}`. Tím získáváme kompletní soubor testovacích bodů s = ([–3; –7], [–3; 0], [–3; 3], $[-\sqrt{5}; 0]$, [0; 0] $[\sqrt{5}; 0]$, [3; –3], [3; 0], [3; 7]), který můžeme použít jako cylindrický algebraický vzorek.

Za povšimnutí též stojí, že zatímco buněk rozkladu D je dohromady 16, testovacích bodů máme jen devět. K tomu došlo proto, že některé sousední buňky, na nichž nedochází ke změně znaménka hodnoty polynomu, jsou sloučeny a reprezentuje je pouze jeden testovací bod. Kupříkladu buňky D_{12}, D_{22} a D_{14}, jenž tvoří jednu z větví paraboly a pro všechny platí rovnost $x^2 - y^2 + xy - 5 = 0$, znaménko polynomu tedy zůstává ve všech těchto buňkách stejné, jsou reprezentovány pouze jedním bodem $[-\sqrt{5}; 0]$.

Toho lze vhodně využít a snížit tak počty testovacích bodů. Pokud místo dvou oddělených příkazů **SemialgebraicComponentInstances** pro $x^2 - y^2 + xy - 5 = 0$ a $x^2 - y^2 + xy - 5 < 0$ zadáme jediný příkaz ve tvaru **SemialgebraicComponentInstances [x^2 – y^2 + x*y – 5 <= 0, {x, y}]**, obdržíme od programu na výstupu pouze tři testovací body $\{\{x \to 0, y \to 0\}, \{x \to -\sqrt{5}, y \to 0\}, \{x \to \sqrt{5}, y \to 0\}\}$. Můžeme takto sestrojit nejmenší možný cylindrický algebraický vzorek s' = ([–3; 0], $[-\sqrt{5}; 0]$, [0; 0] $[\sqrt{5}; 0]$, [3; 0]), pro který platí, že testovací body [–3; 0] a [3; 0] odpovídají buňkám D_{13} a D_{53}, bod $[-\sqrt{5}; 0]$ reprezentuje jednu větev hyperboly sestávající z buněk D_{12}, D_{14} a D_{22}, zatímco bod $[\sqrt{5}; 0]$ odpovídá sjednocení buněk D_{42}, D_{52} a D_{54} a poslední testovací bod [0; 0] odpovídá sjednocení všech zbývajících buněk rozkladu D_{11}, D_{15}, D_{21}, D_{23}, D_3, D_{41}, D_{43}, D_{51} a konečně D_{55}.

Grafické znázornění všech bodů testovacího vzorku s' pak vypadá následovně.



Obrázek 23: Cylindrický algebraický vzorek s' se znázorněnou nerovnicí $x^2 - y^2 + xy - 5 > 0$

Ne vždy ale vyjde soubor testovacích bodů takto relativně jednoduše. Jako ukázku, jak komplikované může být jejich získání, použijme příklad z online nápovědy Wolfram Mathematica 8 Documentation, kde je příkaz předveden na nerovnici $16y^2 + 4x^3 - 4x \leq 1$. Při zadání **SemialgebraicComponentInstances [16y^2 + 4x^3 – 4x <= 1, {x, y}]** vyprodukuje program odpověď ve tvaru

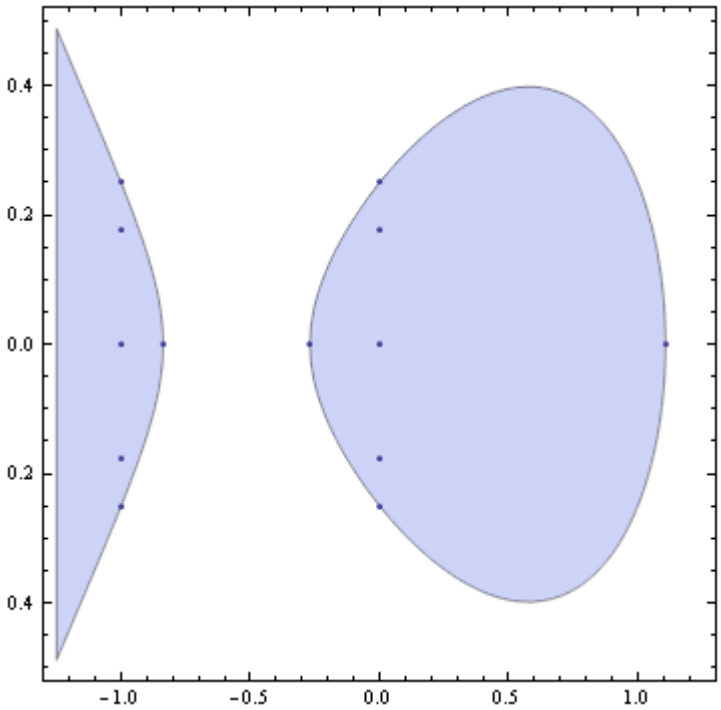
$\{\{x\to -1, y\to -\frac{1}{4}\}, \{x\to -1, y\to 0\}, \{x\to -1, y\to \frac{1}{4}\}, \{x\to -1,$

$y\to -\frac{1}{4\sqrt{2}}\}, \{x\to -1, y\to \frac{1}{4\sqrt{2}}\}, \{x\to 0, y\to -\frac{1}{4}\}, \{x\to 0, y\to 0\}, \{x\to 0, y\to -\frac{1}{4}\}, \{$

$x\to 0, y\to -\frac{1}{4\sqrt{2}}\}, \{x\to 0, y\to \frac{1}{4\sqrt{2}}\}, \{x\to \text{Root}[-1-4\#1+4\#^3\&, 1], y\to 0\}, \{x\to R$

$\text{oot}[-1-4\#1+4\#^3\&, 2], y\to 0\}, \{x\to \text{Root}[-1-4\#1+4\#^3\&, 2], y\to 0\}\}$.

Příkazem **Show [RegionPlot [16y^2 + 4x^3 – 4x <= 1, {x, –0.5, 0.5}, {y, –1.25, 1.25}], ListPlot[{x, y}/.%]]** opět můžeme vykreslit oblasti odpovídající zadané nerovnici i se zjištěným testovacím vzorkem.



Obrázek 24: Testovací body pro nerovnici $16y^2 + 4x^3 - 4x \leq 1$

V interpretaci výstupu se musíme vypořádat s několika komplikacemi. Za prvé se ve výpisu nachází některé body, jejichž souřadnice jsou vyjádřeny pomocí odkazu na kořeny polynomu $4x^3 - 4x - 1$ místo konkrétní hodnotou (v tomto případě jsou kořeny následující $x_1 \doteq -0{,}84$, $x_2 \doteq -0{,}27$, $x_3 \doteq 1{,}11$), za druhé, jak je z obrázku vidět, je testovacích bodů poměrně velký počet, k čemuž dochází částečně díky komplikovanosti zadané nerovnice, ale hlavně kvůli tomu, že v některých oblastech roviny, které člověk na první pohled vnímá jako jedinou buňku, se jich nachází několik. V tomto případě počítač nedokázal provést jejich sloučení tak, aby je bylo možné reprezentovat pouze jediným bodem.

Vezmeme-li k tomu v úvahu, že se jedná pouze o testovací body splňující nerovnici $16y^2 + 4x^3 - 4x \leq 1$, a chceme-li soubor testovacích bodů rozšířit na celý prostor $R[x, y]$ použitím příslušného příkazu i pro nerovnici $16y^2 + 4x^3 - 4x > 1$, je zřejmé, že získávat testovací body pro cylindrickou algebraickou dekompozici tímto způsobem a následně pomocí těchto bodů provádět eliminaci kvantifikátorů je dosti neefektivní. Mathematica naštěstí zvládá rychlejší a efektivnější postup a to prostřednictvím implementovaného algoritmu eliminace kvantifikátorů.

Pro provedení kompletní eliminace kvantifikátorů tak můžeme použít příkaz **Resolve [výraz]** (případně **Resolve [výraz, obor]**, kde oborem mohou být reálná a komplexní čísla nebo hodnoty Booleovy algebry) určený přímo ke zjednodušování

kvantifikovaných matematických formulí. Lze také uplatnit příkaz **Reduce [výraz, x]** (případně **Reduce [výraz, proměnné, obor]**, kde oborem jsou celá, reálná nebo komplexní čísla), obecně užívaný pro řešení či zjednodušení zadaných úloh, jenž lze užít i k odstranění kvantifikátorů. Kombinováním nebo vnořováním těchto výše uvedených příkazů v kombinaci s příkazy **ForAll []** a **Exists []** sestavíme instrukci odpovídající zadané formuli a požadavku, co s ní má program udělat. [WOL12]

7.1.3 Příklady v programu Wolfram Mathematica

Příklad 7.1.3.1: Pomocí programu Mathematica rozhodněte o pravdivosti tvrzení $(\forall x)$: $x^2 \geq 0$.

Jak víme z předchozího textu, toto tvrzení je pravdivé, k tomuto poznání jsme ale došli logickou rozvahou a následně početně, nyní je úkolem provést rozhodnutí pomocí matematického programu. Zapíšeme příkaz ve tvaru **Resolve [ForAll [x, x^2 >= 0]]**, který počítač v podstatě instruuje, aby zjednodušil nerovnici $x^2 \geq 0$, jež má platit pro všechna reálná x, a dostaneme odpověď `True`. To znamená, že formule $(\forall x)$: $x^2 \geq 0$ je pravdivá. ♦

Jak již bylo poznamenáno dříve, je třeba při zápisu příkazů dávat pozor na správnost a přesnost instrukcí. Problémy mohou nastat zvláště při uspořádání kvantifikátorů a kvantifikovaných proměnných ve formuli. Ukažme si to na příkladu.

Příklad 7.1.3.2: V programu Mathematica sestrojte příkaz pro zjednodušení formule $(\exists x)(\forall y)$: $x + y = 0$ a rozhodněte o její pravdivosti.

Zadaná formule $(\exists x)(\forall y)$: $x + y = 0$ říká, že existuje takové reálné číslo x, že pro všechna reálná čísla y pak platí rovnost $x + y = 0$. Takové situaci samozřejmě odpovídá příkaz ve tvaru **Resolve [Exists [x, ForAll [y, x + y == 0]]]**. Zvídavého čtenáře jistě napadne, že dosti podobnou instrukcí je **Resolve [ForAll [y, Exists [x, x + y == 0]]]**, která by snad mohla předchozí příkaz nahradit. Tak tomu ale není, každá z těchto formulací vyjadřuje jinou skutečnost. První skutečně popisuje v zadání danou situaci, kdežto druhá se dá matematickou formulí zapsat jako $(\forall y)(\exists x)$: $x + y = 0$, neboli pro všechna reálná čísla y existuje nějaké reálné číslo x takové, že součet $x + y$ je roven nule.

Rozdíl je pak následně i ve vyhodnocení pravdivosti. Zatímco příkaz **Resolve [ForAll [y, Exists [x, x + y == 0]]]** patřící k formuli $(\forall y)(\exists x)$: $x + y = 0$ dá uživateli odpověď `True` (a tedy formule je vždy pravdivá), neboť pro libovolné reálné číslo

existuje číslo opačné a jejich součet je roven nule, na příkaz **Resolve [Exists [x, ForAll [y, x + y == 0]]]**, který odpovídá zadanému tvrzení, odpoví počítač `False`. Tuto odpověď budeme interpretovat tak, že zadaná formule $(\exists x)(\forall y)$: $x + y = 0$ není pravdivá, jelikož žádné takové reálné číslo x, jehož součet s jiným libovolným reálným číslem by byl nulový, neexistuje. ♦

Příklad 7.1.3.3: Určete obor pravdivosti nerovnice $x^4 + 2x^3 - 21x^2 - 22x + 40 < 0$.

Pro začátek zapišme zadanou nerovnici o jedné neznámé pomocí tvrzení $(\exists x)$: $x^4 + 2x^3 - 21x^2 - 22x + 40 < 0$, ta říká, že existuje takové reálné číslo x, že výsledná hodnota při jeho dosazení do tohoto polynomu bude menší než nula. Užití příkazu ve steném tvaru jako v přechozích příkladech, a sice **Resolve [Exists [x, x^4 + 2x^3 – 21x^2 – 22x + 40 < 0]]**, mnoho nepomůže, dostaneme totiž odpověď `True`, která nás pouze informuje, že takové číslo x existuje.

Můžeme ale zkusit trochu experimentovat se zápisem příkazu, zapišme jej ve tvaru **Resolve [ForAll [x, x0 < x < x1, x^4 + 2x^3 – 21x^2 – 22x + 40 < 0], {x0, x1}]**, tedy že nás zajímají všechna čísla x patřící do jistého intervalu $(x_0; x_1)$. Odpovědí je řetězec

`(x0<–5&&x1≤x0)||(–5≤x0≤–2&&x1≤–2)||(–2<x0<1&&x1≤x0)||(1≤x0≤4&&x1≤4)||(x0>4&&x1≤x0)`. Na první pohled je interpretace těchto výsledků obtížná, při bližším pohledu ale zjistíme, že můžeme „vyškrtnout“ první, třetí a pátou závorku, které obsahují nerovnost `x1≤x0`, jenž odporuje v příkazu zadané nerovnosti `x0<x<x1`. Tak zůstane pouze zápis `(–5≤x0≤–2&&x1≤–2)||(1≤x0≤4&&x1≤4)`, z něhož již dostaneme sjednocení intervalů $(-5; -2) \cup (1; 4)$, v nichž nerovnost $x^4 + 2x^3 - 21x^2 - 22x + 40 < 0$ skutečně platí. ♦

Příklad 7.1.3.4: Rozhodněte o pravdivosti formule $(\forall y)$: $y^3 - 3y^2 + 2y - 6 \geq 0$. V případě, že se jedná o nesplnitelnou formuli, pokuste se nalézt všechny hodnoty y, kdy platí nerovnost $y^3 - 3y^2 + 2y - 6 \geq 0$.

Zadáme příkaz ve tvaru **Resolve [ForAll [y, y^3 – 3y^2 + 2y – 6 >= 0]]** a dostaneme odpověď `False`. Jedná se totiž o polynom třetího stupně a je tedy jisté, že pro některé hodnoty, kterých může nabývat reálná proměnná y, je výsledek záporný, zatímco pro jiné naopak kladný. Zadaná formule tedy neplatí.

Zapišme příkaz takto: **Resolve [ForAll [y, y > y0, y^3 – 3y^2 + 2y – 6 >= 0]]**; tentokrát dostaneme odpověď `y0≥3`. Z toho vidíme, že zadaná nerovnost je splněna pro libovolnou hodnotu čísla $y \geq 3$. Napíšeme-li naopak příkaz **Resolve [Exists [y, y < 3, y^3 – 3y^2 + 2y – 6 >= 0]]**, výsledek bude `False`, což znamená, že pro žádné $y < 3$ uvedená nerovnost neplatí.

Kromě toho, že se nám podařilo eliminovat kvantifikátory ve formuli $(\forall y)$: $y^3 - 3y^2 + 2y - 6 \geq 0$, zjistili jsme i několik informací o samotném polynomu $y^3 - 3y^2 + 2y - 6$: polynom má pouze jeden reálný kořen $y = 3$, pro $y > 3$ nabývá kladných hodnot, zatímco pro $y < 3$ dostáváme hodnoty záporné. ♦

Výše v textu jsme se již zmínili o tom, že eliminaci kvantifikátorů lze uplatnit i v takových oblastech, jakými jsou například analytická geometrie, vyšetřování parametrických systémů či práce s limitami nebo omezeností funkcí. Ukažme si to na konkrétních příkladech:

Příklad 7.1.3.5: Nalezněte řešení rovnice $x^5 - 2x^2 + 3 = -x^3 + 2x + 3$.

Podle zadání hledáme takové hodnoty proměnné x, pro něž by platila rovnost $x^5 - 2x^2 + 3 = -x^3 + 2x + 3$. Jinými slovy nás může předně zajímat, je-li pravdivá matematická formule ve tvaru $(\exists x)$: $x^5 - 2x^2 + 3 = -x^3 + 2x + 3$.

Zadejme příslušný příkaz **Resolve [Exists [x, x^5 – 2x^2 + 3 == –x^3 + 2x + 3]]**. Odpověď je `True`, alespoň jeden kořen rovnice tedy existuje. Podle zadání ale máme řešení přímo nalézt, nejen pouze ověřit, zda existuje.

Vzhledem k tomu, že rovnice obsahuje pouze jednu neznámou, která je však v zápisu příslušné matematické formule kvantifikována, dojde při eliminaci kvantifikátorů k jejímu odstranění. Hodnotu kořenů tak nejde zjistit.

Proveďme ale drobnou úpravu. Zaveďme druhou neznámou y, přičemž ji položíme rovnou například pravé straně naší rovnosti. Dostáváme tak řetězec rovností $y = x^5 - 2x^2 + 3 = -x^3 + 2x + 3$, který můžeme též přepsat do tvaru soustavy dvou rovnic o dvou neznámých

$$y = x^5 - 2x^2 + 3$$
$$x^5 - 2x^2 + 3 = -x^3 + 2x + 3.$$

Kvantifikujme existenčním kvantifikátorem pouze jednu z neznámých, řekněme, že jí bude neznámá y, a formule pak bude ve tvaru $(\exists y)$: $y = x^5 - 2x^2 + 3 \wedge x^5 - 2x^2 + 3 = -x^3 + 2x + 3$. Do programu pak tento zápis můžeme zadat spolu s příkazem pro její vyřešení jako **Resolve [Exists [y,**

y == x^5 – 2x^2 + 3 && x^5 – 2x^2 + 3 == – x^3 + 2x + 3]][12]. Výstupem je ekvivalentní formule `x==0||-2-2x+x^2+x^4==0`.

Jedním kořenem je pak $x_1 = 0$, ostatní řešení odpovídají reálným kořenům rovnice $x^4 + x^2 - 2x - 2 = 0$, které jsou dva, na jejich hodnoty však tímto přístupem uživatel nedosáhne.[13]

Nalezení zbývajících kořenů je pak tedy záležitostí užití jiných příkazů programu, například **FindRoot [výraz, {x, x0}]**, které však již s eliminací kvantifikátorů přímo nesouvisí. ♦

Příklad 7.1.3.6: Zjistěte, zda mají kuželosečky dané rovnicemi $x^2 + y^2 = 4$ a $y^2 = x + 1$ nějaký průsečík či nikoliv.

Máme dvě kuželosečky, kružnici zadanou rovnicí $x^2 + y^2 = 4$ (střed má v bodě $S = [0; 0]$ a poloměr $r = 2$) a parabolu $y^2 = x + 1$ (vrchol $V = [-1; 0]$, ohnisko $F = \left[-\frac{3}{4}; 0\right]$ a řídící přímka $d : x = -\frac{5}{4}$). Pokud existuje nějaký jejich společný bod, pak musí být také splněna formule $(\exists x, y)$: $x^2 + y^2 - 4 = 0 \wedge x - y^2 + 1 = 0$. Zapišme příkaz **Resolve [Exists [{x, y}, x^2 + y^2 – 4 == 0 && x – y^2 + 1 == 0]]** a nechme program provést eliminaci. Odpověď je `True`, takže taková dvojice (je minimálně jedna) čísel x a y, která splňuje zadanou formuli, existuje. Tím jsme dokázali, že daná kružnice $x^2 + y^2 = 4$ a parabola $y^2 = x + 1$ mají alespoň jeden průsečík.

Zkusme s příkladem trochu experimentovat a upravme počítačový příkaz do tvaru **Resolve [Exists [x, x^2 + y^2 – 4 == 0 && x – y^2 + 1 == 0]]**. Vzhledem k tomu, že tentokrát máme kvantifikovánu pouze proměnnou x, nikoliv však proměnnou y, bude se odpověď lišit: `-3-y^2+y^4==0`. Opět jsme dostali formuli bez kvantifikátorů, přičemž víme, že reálná řešení rovnice $y^4 - y^2 - 3 = 0$ budou odpovídat y-ovým souřadnicím dokazovaných průsečíků. Nyní musíme zjistit, jestli tyto reálné kořeny skutečně existují. Na příkaz **Resolve [Exists [y, y ∈ Reals, y^4 – y^2 – 3 == 0]]** nám dá program odpověď `True`, tím jsme dospěli ke

---

[12] Můžeme také využít tranzitivnosti relace „rovná se“ a celý příkaz zapsat ve tvaru **Resolve [Exists [y, y == x^5 – 2x^2 + 3 == – x^3 + 2x + 3]]**, výsledek eliminace bude v obou případech totožný.

[13] Bohužel zde nepomůže ani užití příkazu **SemialgebraicComponentInstances**, jak by se mohlo na první pohled zdát, program se i v tomto případě odkáže na kořeny polynomu $x^4 + x^2 - 2x - 2$.

stejnému výsledku jako u předchozího postupu a dokázali jsme, že kružnice $x^2 + y^2 = 4$ a parabola $y^2 = x + 1$ skutečně mají alespoň jeden průsečík. ♦

Příklad 7.1.3.7: Vyšetřete existenci řešení kvadratické rovnice $2 \cdot x^2 + b \cdot x + 3 < 0$ v závislosti na zadaném parametru *b*.

Prvním a zároveň nejjednodušším úkolem v tomto případě bude zjistit, zda nějaké řešení vůbec existuje. Použijme příkaz **Resolve [Exists [{x, b}, 2x^2 + b*x + 3 < 0]]**, na nějž dostaneme od počítače odpověď `True`. Existuje tedy taková dvojice *x*, *b*, pro niž je splněna daná nerovnost.

Zajímají-li uživatele další podrobnosti, je třeba zadaný zápis poupravit. Použijeme příkaz **Resolve [Exists [x, 2x^2 + b*x + 3 < 0]]**, který odpovídá formuli $(\exists x)$: $2x^2 + bx + 3 < 0$, výstupní formule bez kvantifikátorů potom bude obsahovat pouze parametr *b* a bude ve tvaru `b<–2√6 || b>2√6`.

Výsledek si zaslouží trochu bližší prozkoumání, výpis `b<–2√6 || b>2√6` uživatele informuje o tom, že skutečně existuje hodnota proměnné *x*, pro kterou nerovnost platí, ovšem tato existence je závislá právě na hodnotě parametru *b*, která musí spadat do sjednocení intervalů $(-\infty; -2\sqrt{6}) \cup (2\sqrt{6}; \infty)$. Jinými slovy, je-li parametr *b* hodnota z výše zmíněného sjednocení intervalů, pak má nerovnice $2x^2 + bx + 3 < 0$ reálné řešení. ♦

Příklad 7.1.3.8: Určete limitu funkce $f(x) = \dfrac{5x}{3x-2}$ pro hodnoty $x \to \infty$.

Zjištění limity racionálně lomené funkce $f(x) = \dfrac{5x}{3x-2}$ pro $x \to \infty$ je poměrně jednoduché $(\lim\limits_{x\to\infty} \dfrac{5x}{3x-2} = \dfrac{5}{3})$, i přesto nám tento příklad může posloužit k netradičnímu pohledu na problém. Popišme jej pomocí ekvivalentního zápisu definice limity: $(\forall \varepsilon > 0)(\exists x_0 \in R)(\forall x > x_0)$: $\dfrac{5}{3} - \varepsilon < \dfrac{5x}{3x-2} < \dfrac{5}{3} + \varepsilon$. Zadáme-li počítači příkaz **Resolve [ForAll [ε, ε > 0, Exists [x0, x0 ∈ Reals, ForAll [x, x > x0, 5/3 – ε < 5x/(3x – 2) < 5/3 + ε]]]]**, vypíše výsledek eliminace `True`. Víme tedy, že náš kvantifikovaný zápis limity funkce $f(x)$ je správný a pravdivý. Limita $\lim\limits_{x\to\infty} \dfrac{5x}{3x-2}$ je skutečně rovna $\dfrac{5}{3}$.

Nemusíme se ale omezovat pouze na potvrzování námi zjištěných faktů, počítač nám může limitu i sám spočítat: po zadání **Resolve [ForAll [ε, ε > 0, Exists [x0, ForAll [x, x > x0 && a ∈ Reals, a – ε < 5x/(3x – 2) < a + ε]]], {a}]** dostáváme výsledek (hledanou limitu funkce) `a==5/3.`♦

Příklad 7.1.3.9: Určete, zda je funkce $f(x) = \frac{5x^2}{3x-2}$ omezená ve svém definičním oboru.

Nejprve ověřme omezenost shora. I v tomto příkladu musíme nejdříve problém převést na kvantifikovanou formuli, kterou tentokrát můžeme zapsat $(\forall K > 0)(\exists x)$: $\frac{5x^2}{3x-2} > K$. Počítači ji zadáme jako příkaz **Resolve [ForAll [K, K > 0, Exists [x, 5x^2/(3x – 2) > K]]]** a odpovědí bude `True`. To znamená, že funkce $f(x) = \frac{5x^2}{3x-2}$ není shora omezená. Ověřme ještě omezenost zdola pomocí formule $(\forall K < 0)(\exists x)$: $\frac{5x^2}{3x-2} < K$ a příkazu **Resolve [ForAll [K, K < 0, Exists [x, 5x^2/(3x – 2) < K]]]**. I tentokrát dostaneme odpověď `True`, funkce $f(x)$ tedy není omezená ani zdola.♦

[HON08, HOR09, HOP, WOL12]

Poznámka: Ekvivalentně jde příslušná formule zapsat i jinak. První pro omezenost shora můžeme nahradit formulí ve tvaru $(\exists K)(\forall x)$: $\frac{5x^2}{3x-2} < K$, druhou pro omezenost zdola pak zápisem $(\exists K)(\forall x)$: $\frac{5x^2}{3x-2} > K$.♦

Zároveň nám eliminace kvantifikátorů může pomoci při řešení jednoduchých optimalizačních úloh, v takovém případě je pak zvlášť nutné zapsat danou formuli přesně, přičemž se pravděpodobně nevyhneme ani dodatečným podmínkám z dané úlohy vyplývající. Ukažme si to na následující úloze.

Příklad 7.1.3.10: Uhlí z uhelných dolů Karvinska a Mostecka se vozí do Olomouce, Českých Budějovic a Brna. Denně se může z Karvinska vyvézt 300 tun uhlí a z Mostecka 250 tun. Denně je potřeba dodat 140 tun uhlí do Olomouce, do Českých Budějovic 200 tun a do Brna 210 tun. Nalezněte způsob rozvozu, při kterém budou náklady nejnižší, jestliže víte, že nejvyšší možná denní dodávka z Karvinska do

Brna je kvůli výluce snížena na maximálně 120 tun uhlí. Náklady na dopravu jedné tuny uhlí z dolu do místa spotřeby jsou v tabulce.

cena v € za tunu	**Olomouc**	**Č. Budějovice**	**Brno**
Karvinsko	12	12	6
Mostecko	8	7	9

(Úloha je založena na optimalizačních úlohách z [DUP82], kde lze nalézt i tradiční způsob řešení dané problematiky.)

Než začneme provádět samotnou eliminaci, je nutné nejprve sestavit příslušnou matematickou formuli. Tomu v tomto případě předchází ještě zavedení neznámých a jisté úpravy.

Podle zadání sestavíme seznam vypovídající o celkových možných denních vývozech z těžebních oblastí a celkových dodávkách do cílových měst.

z Karvinska ..... 300 tun
z Mostecka ..... 250 tun
do Olomouce ..... 140 tun
do Českých Budějovic ... 200 tun
do Brna ..... 210 tun

Dále označme počet tun uhlí přepravených z Karvinska do Olomouce neznámou x a počet tun dovezených též z Karvinska tentokrát do Českých Budějovic neznámou y. Pak je jasné, že počet tun dovezených z Karvinska do Brna odpovídá rozdílu $300 - (x + y)$.

Získáváme tak první řádek tabulky s daty o přepraveném uhlí.

přepravené tuny	**Olomouc**	**Č. Budějovice**	**Brno**
Karvinsko	x	y	$300 - (x + y)$

Obdobným způsobem jme schopni doplnit údaje i do druhého řádku týkajícího se přepravy uhlí z Mostecka. Do Olomouce je nutné dopravit celkem 140 tun uhlí, z Mostecka je jich tedy potřeba přivézt ještě $140 - x$, do Českých Budějovic již bylo dovezeno y tun z Karvinska, z Mostecka tak zbývá dopravit $200 - y$ a konečně z Mostecka do Brna bude dovezeno $210 - [300 - (x + y)] = x + y - 90$

přepravené tuny	**Olomouc**	**Č. Budějovice**	**Brno**
Karvinsko	x	y	$300 - (x + y)$
Mostecko	$140 - x$	$200 - y$	$x + y - 90$

Vzhledem k tomu, že pracujeme s množstvím (tunami uhlí), musí být jednotlivé výše vypsané výrazy nezáporné. Pro první řádek tabulky tedy platí podmínky

$$x \geq 0,\ y \geq 0,\ x + y \leq 300,$$

pro druhý pak

$$x \leq 140,\ y \leq 200 \text{ a } x + y \geq 90.$$

Zároveň ještě musíme zohlednit poslední podmínku obsaženou přímo v zadání a říkající, že kvůli výluce lze každý den dopravit z Karvinska do Brna maximálně 120 tun, proto $300 - (x + y) \leq 120$ a po úpravě $x + y \geq 180$.

Nakonec sestavíme funkční předpis pro celkové zisky za přepravené uhlí $f(x, y) = 12x + 12y + 6\cdot[300 - (x + y)] + 8\cdot(140 - x) + 7\cdot(200 - y) + 9\cdot(x + y - 90)$.

Tento předpis ještě upravíme do tvaru $f(x, y) = 7x + 8y + 3510$.

Úloha nyní spočívá v hledání extrému, jmenovitě minima, pro funkci $f(x, y)$ s předpisem odpovídajícím polynomiálnímu výrazu ve dvou proměnných, přičemž obě proměnné jsou omezeny stanovenými podmínkami.

Zda nějaké řešení vůbec existuje, můžeme ověřit eliminováním kvantifikátorů z formule $(\exists\, x_0, y_0,\ x_0 \geq 0,\ y_0 \geq 0,\ x_0 + y_0 \leq 300,\ x_0 \leq 140,\ y_0 \leq 200,\ x_0 + y_0 \geq 90,\ x_0 + y_0 \geq 180)(\forall\, x, y,\ x \geq 0,\ y \geq 0,\ x + y \leq 300,\ x \leq 140,\ y \leq 200,\ x + y \geq 90,\ x + y \geq 180)$: $7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$.[14] Tato formule říká, že v podmínkami vymezené části prostoru existuje nějaký bod o souřadnicích $[x_0; y_0]$ takový, že jeho funkční hodnota je menší než pro jakýkoliv jiný bod z této oblasti.

Zadejme do programu Mathematica odpovídající příkaz **Resolve [Exists [{x0, y0}, x0 ≥ 0 && y0 ≥ 0 && x0 + y0 ≤ 300 && x0 ≤ 140 && y0 ≤ 200 && x0 + y0 ≥ 90 && x0 + y0 ≥ 180, ForAll [{x, y}, x ≥ 0 && y ≥ 0 && x + y ≤ 300 && x ≤ 140 && y ≤ 200 && x + y ≥ 90 && x + y ≥ 180, 7x + 8y + 3510 ≥ 7x0 + 8y0 + 3510]]]**. Odpověď programu je `True`, to lze interpretovat jako existenci řešení. Skutečně tedy existuje alespoň jedna taková uspořádaná dvojice $[x_0; y_0]$, která produkuje hodnotu $f(x, y)$ menší či nejvýše rovnou funkčním hodnotám všech ostatních dvojic $[x; y]$.

V této chvíli by bylo příjemné moci zjistit přímo hodnoty těchto dvou neznámých x_0 a y_0. Zkusme proto sestavit poněkud obměněnou matematickou formuli $(\forall\, x, y,$

[14] Z formule můžeme bez problému vypustit podmínky $x_0 + y_0 \geq 90$ a $x + y \geq 90$, protože ještě silnější podmínkou jsou podmínky $x_0 + y_0 \geq 180$ a $x + y \geq 180$.

$x \geq 0$, $y \geq 0$, $x + y \leq 300$, $x \leq 140$, $y \leq 200$, $x + y \geq 90$, $x + y \geq 180$): $7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$. V ní se vyskytují dvě kvantifikované proměnné x, y a dvě volné proměnné x_0, y_0. Důsledkem eliminace kvantifikátorů v této formuli tak bude vyjádření formule ekvivalentní pomocí právě těchto volných proměnných.

Na příkaz **Resolve [ForAll [{x, y}, x ≥ 0 && y ≥ 0 && x + y ≤ 300 && x ≤ 140 && y ≤ 200 && x + y ≥ 90 && x + y ≥ 180, 7x + 8y + 3510 ≥ 7x0 + 8y0 + 3510]]** program odpoví ekvivalentní formulí tvaru `(x0|y0)∈Reals&& −1300+7x0+8y0≤0`.

Řešení $[x_0; y_0]$ podle výsledku leží v polorovině p: $7x + 8y - 1300 \leq 0$, není však takto jednoznačně dáno, protože ve vstupní formuli nebyly zohledněny podmínky týkající se hodnot x_0 a y_0. Dalším možným postupem by v takovém případě bylo hledání průniku poloroviny p a oblasti omezené zmíněnými podmínkami, to je však zbytečně složité. Jednodušší cestu zvolíme, pokusíme-li se z výstupní formule odstranit jednu z proměnných. Toho lze poměrně lehce dosáhnout i tím, že pozměníme vstupní formuli do podoby, v níž se bude vyskytovat pouze jedna volná proměnná, řekněme y_0.

Taková vstupní formule bude vypadat takto ($\exists\, x_0,\ x_0 \geq 0,\ y_0 \geq 0,\ x_0 + y_0 \leq 300,\ x_0 \leq 140,\ y_0 \leq 200,\ x_0 + y_0 \geq 90,\ x_0 + y_0 \geq 180$)($\forall\, x, y,\ x \geq 0,\ y \geq 0,\ x + y \leq 300,\ x \leq 140,\ y \leq 200,\ x + y \geq 90,\ x + y \geq 180$): $7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$. Příkaz programu zapíšeme do formy **Resolve [Exists [x0, x0 ≥ 0 && y0 ≥ 0 && x0 + y0 ≤ 300 && x0 ≤ 140 && y0 ≤ 200 && x0 + y0 ≥ 90 && x0 + y0 ≥ 180, ForAll [{x, y}, x ≥ 0 && y ≥ 0 && x + y ≤ 300 && x ≤ 140 && y ≤ 200 && x + y ≥ 90 && x + y ≥ 180, 7x + 8y + 3510 ≥ 7x0 + 8y0 + 3510]]]** a výsledná formule bez kvantifikátorů již je ve tvaru jednoduché rovnosti `y0==40`.

Následné zjištění hodnoty $x_0 = 140$ můžeme provést dopočítáním z hodnoty y_0, případně je možné upravit vstupní formuli tak, aby volnou proměnnou byla x_0 a kvantifikovanou y_0.

Pro úplnost ještě vyjádřeme výsledek celého příkladu, který je nejpřehlednější v tabulce, přičemž celkové náklady na přepravu uhlí v této situaci jsou 4810 euro:

přepravené tuny	**Olomouc**	**Č. Budějovice**	**Brno**
Karvinsko	140	40	120
Mostecko	0	160	90

Na závěr úlohy ještě dodejme čtyři poznámky, první z nich není nikterak zásadní, další tři už mají podstatnější dopad.

Poznámka 1: Přestože jsme dospěli k celočíselným výsledkům, nebyla by v tomto případě problematická ani interpretace výsledku v podobě desetinných čísel, a to vzhledem k tomu, že se jedná o jednotky hmotnosti, které lze dále dělit. Obecně to však v optimalizačních úlohách vždy neplatí.♦

Poznámka 2: Jak jsme si mohli všimnout, je úloha postavena tak, že „uhelná nabídka“ odpovídá „uhelné poptávce“ (celkem je z míst těžby vyvezeno 550 tun a stejné množství je také spotřebováno), pokud by tomu tak nebylo a například nabídka by převyšovala poptávku, bylo by řešení komplikovanější (například hodnoty v posledním sloupečku tabulky by tímto způsobem nebylo možné vyjádřit konkrétním výrazem, ten by zde fungoval pouze jako jakási horní mez).♦

Poznámka 3: V průběhu zjišťování výsledku jsme s velkou výhodou využili toho, že program Mathematica umožňuje zadat příkazy reprezentující oba kvantifikátory ve tvarech **Exists**/**ForAll [proměnná, podmínka, výraz]**. Tím pádem je možné zadat kvantifikované proměnné spolu s podmínkami, které se jich týkají, a jak vzniklá matematická formule, tak i její zápis jsou relativně jednoduché oproti případu, kdy bychom se museli obejít bez této možnosti a pracovat pouze s předpisy ve tvaru **Exists**/**ForAll [proměnná, výraz]**. Na způsob řešení takové situace se podíváme v kapitole *6.2.3 Příklady v programu QEPCAD B*, který nemá takovéto možnosti zápisu podmínek jako Wolfram Mathematica.♦

Poznámka 4: K řešení lze přistupovat i tak, že nebudeme na jeho počátku provádět úpravy spočívající ve vyjádření všech dotazovaných hodnot pouze dvěma neznámými. Je totiž možné každou neznámou hodnotu (tedy hmotnost uhlí přepravenou z místa těžby do místa spotřeby) označit jinou neznámou, přičemž tabulka přepravených tun by vypadala následovně:

přepravené tuny	**Olomouc**	**Č. Budějovice**	**Brno**
Karvinsko	u	v	w
Mostecko	x	y	z

Podle toho by se samozřejmě změnila i soustava podmínek a příslušná účelová funkce:

$$u \geq 0,\ v \geq 0,\ w \geq 0,\ x \geq 0,\ y \geq 0,\ z \geq 0,$$

$$u + v + w \leq 300,\ x + y + z \leq 250,\ u + x \geq 140,\ v + y \geq 200,\ w + z \geq 210$$

a

$$f = 12u + 12v + 6w + 8x + 7y + 9z.$$

Toto svým způsobem přirozenější vyjádření je sice na první pohled přehlednější, avšak je nutné si uvědomit, že tentokrát odpadá možnost jakéhokoliv grafického znázornění, jelikož se nyní nacházíme v šestirozměrném prostoru. Přitom zápis příslušné kvantifikované formule bude též komplikovanější.

I přesto by hledání ekvivalentní formule bez kvantifikátorů neznamenalo žádný velký problém, jelikož využití cylindrické algebraické dekompozice je při vyšším počtu rozměrů limitováno pouze časově. ♦

Při pohledu na předchozí příklad je jasné, že vzhledem k tomu, že funkční vyjádření $f(x, y)$ i křivky vymezující danou část prostoru podle zavedených podmínek, jsou v obou proměnných lineární, lze hledané minimum (resp. extrém) očekávat právě na některé z hraničních přímek[15]. V nejlepším případě extrém leží v průsečíku omezujících přímek (jde o situaci, kdy je právě jedno řešení úlohy), v ostatních případech celá omezující přímka splyne s množinou bodů s minimální funkční hodnotou, (tehdy je řešení dokonce nekonečně mnoho, jako řešení celé slovní úlohy však můžeme přijmout jen ta, která mají rozumnou interpretaci do reálného světa).

Zobecníme-li zadání, nemusíme se setkat pouze s lineárním funkčním předpisem, který zkoumáme, a s lineárními omezujícími křivkami. Ty ve výsledku nemusí existovat vůbec.

Příklad 7.2.3.11: Nalezněte globální extrém funkce $f(x, y) = 2x^2 + y^2 - 2y + 5$.

Zadaná funkce $f(x, y)$ popisuje paraboloid, nalezení jeho extrému – vrcholu, se jeví jako netriviální záležitost, při níž se obecně uplatňuje diferenciální počet. Zkusme však vyjít z matematické formule podobné té, kterou jsme již použili v příkladu 7.1.3.9. Rozdíl je pouze v tom, že tentokrát pracujeme se dvěma proměnnými místo jedné.

Zkusme nejprve zjistit, zda je zadaná funkce shora nebo zdola omezená. Pokud tomu tak bude, pokusíme se nalézt přímo extrém.

Pro ověření omezenosti shora použijeme formuli $(\exists K)(\forall x, y)$: $2x^2 + y^2 - 2y + 5 < K$, v přepisu pro program Mathematica bude příkaz vypadat takto **Resolve [Exists K,**

[15] Speciálním případem by byla úloha, kde by vzniklá funkce $f(x, y)$ byla na celé vymezené (případně i neomezené) oblasti konstantní a řešením by byla jakákoliv uspořádaná dvojice $[x, y]$ z této (ne)omezené oblasti.

[ForAll [{x, y}, 2x^2 + y^2 – 2y + 5 < K]]. Odpovědí je výstup `False`, funkce tedy není shora omezená a nemá cenu hledat maximum. Naproti tomu eliminování kvantifikátorů z formule $(\exists K)(\forall x, y)$: $2x^2 + y^2 - 2y + 5 > K$ příkazem **Resolve [Exists K, [ForAll [{x, y}, 2x^2 + y^2 – 2y + 5 > K]]]** přinese výstupní formuli `True`. To znamená, že existuje nějaké reálné číslo K, jenž je menší než jakákoliv funkční hodnota $f(x, y)$, a má smysl hledat minimum.

Právě použité formule obsahují tři proměnné, přičemž všechny jsou kvantifikované. Jelikož všechny kvantifikované proměnné při provedení eliminace kvantifikátorů z formule zmizí, dostali jsme odpovědi pouze `True` nebo `False`. Snížíme-li ale počet kvantifikátorů a některá z proměnných bude v roli volné proměnné, zůstane v také ve výstupní formuli bez kvantifikátorů.

Zapišme příkaz **Resolve [ForAll [{x, y}, 2x^2 + y^2 – 2y + 5 > K]]**. Proměnná K nyní není kvantifikována, takže ve výstupní formuli `K∈Reals&&–4+K<0` zůstane a nám se dostává informace o tom, jakých hodnot může nabývat, aby zadaná matematická formule platila. Úpravou nerovnosti $-4 + K < 0$ dostaneme nerovnost $K < 4$ a víme, že jakékoliv reálné číslo K splňující tuto podmínku pak splňuje i zadanou formuli. Neboli hodnota 4 je funkční hodnota minima funkce $f(x, y)$.

Určení samotné funkční hodnoty ale nemusí být dostačující, uživatele navíc pravděpodobně budou zajímat u hodnoty x a y, v nichž funkce svého minima nabývá.

Můžeme tedy stejně jako v předchozí optimalizační úloze využít zavedení nekvantifikovaných proměnných x_0 a y_0, které po eliminaci kvantifikátorů ve výsledné formuli zůstanou. Eliminujme tedy z formule $(\forall x, y)$: $2x_0^2 + y_0^2 - 2y_0 + 5 > 2x^2 + y^2 - 2y + 5$ proměnné x a y příkazem **Resolve [ForAll [{x, y}, 2x0^2 + y0^2 – 2y0 + 5 <= 2x^2 + y^2 – 2y + 5]]**. Výsledkem je formule bez kvantifikátorů `x0∈Reals&&y0>0&&2x0^2–2y0+y0^2≤–1`.

Víme tedy, že řešením je uspořádaná dvojice reálného x_0 a reálného kladného y_0, pro něž platí nerovnice $2x_0^2 + y_0^2 - 2y_0 \leq -1$.

Její řešení není komplikované, stačí provést doplnění na čtverec a dostaneme nerovnici ve tvaru $2x_0 + (y_0 - 1)^2 \leq 0$, jejímž řešením je bod o souřadnicích [0; 1].

Ke stejnému řešení ale dojdeme i bez takovéto úvahy při použití eliminace na matematickou formuli $(\exists x_0)(\forall x, y)$: $2x_0^2 + y_0^2 - 2y_0 + 5 > 2x^2 + y^2 - 2y + 5$.

V takovém případě je výsledkem zadaného příkazu **Resolve [Exists [x0, ForAll [{x, y}, 2x0^2 + y0^2 – 2y0 + 5 <= 2x^2 + y^2 – 2y + 5]]]** formule `y0==1`. Analogicky lze nahrazením kvantifikované proměnné x_0 proměnnou y_0 obdržet výstup `x0==0`.

Souřadnice bodu, v němž má funkce $f(x, y)$ minimum, jsou [0; 1]. To lze ověřit i použitím příkazu **FindMinimum [2x^2 + y^2 – 2y + 5, {x, y}]**, který produkuje výstup ve tvaru `{4.,{x→0.,y→1.}}`. Požíváním tohoto příkazu bychom se ovšem již dostali daleko od samotné eliminace kvantifikátorů. ♦

Zajímavým aspektem zmíněným již v úvodu je zjištění strojového času potřebného pro požadovaný výpočet. V programu Mathematica je k jeho zjištění používán příkaz **Timing [výraz]**, přičemž jeho argument odpovídá příkazu, jehož časová náročnost má být vyhodnocena.

7.2 QEPCAD

7.2.1 Program QEPCAD a QEPCAD B

Samotné označení QEPCAD je zkratkou anglického Quantifier Elimination by Partial Cylindrical Algebraic Decomposition, což poskytuje informaci o tom, jakou metodou program eliminaci kvantifikátorů provádí. Využívá při tom částečnou cylindrickou algebraickou dekompozici.

Původním autorem aplikace je Hoon Hong, profesor v oboru symbolických výpočtů na Státní univerzitě Severní Karolíny (North Carolina State University). V průběhu času do vývoje programu vstupovali další přispěvatelé, včetně například George E. Collinse, a v současné době je nejdostupnější (z hlediska dostupnosti instalačních souborů a uživatelské dokumentace; obojí lze nalézt na webové adrese http://www.usna.edu/cs/~qepcad/B/QEPCAD.html) verze programu QEPCAD B, na jejímž dalším vývoji v současné době pracuje Christopher W. Brown z katedry informatiky Námořní akademie Spojených států (United States Naval Academy). [BRO02, BRO03]

7.2.2 Příkazy programu QEPCAD B

Spuštění i obsluha programu QEPCAD probíhá prostřednictvím okna příkazového řádku, do nějž jsou programem vypisovány též výsledky. Po spuštění musí uživatel nejprve zadat název prováděné operace vepsaný do hranatých závorek (jde jen o

neformální označení, není tedy třeba vymýšlet nic složitého), následuje vložení seznamu použitých proměnných v kulatých závorkách uspořádaný tak, že volné proměnné jsou na prvních pozicích seznamu, kvantifikované proměnné na jeho konci, dále je potřeba zadat počet volných proměnných a nakonec kvantifikovanou vstupní formuli.

Tu je nutné vypsat ve tvaru `(Q1 x1)...(Qn xn)[T1...Tm].`, kde Q_i jsou kvantifikátory, x_i jsou příslušné kvantifikované proměnné a T_j jsou jednotlivá tvrzení spojená logickými spojkami, přičemž $1 \leq i \leq n$ a $1 \leq j \leq m$. Celá takto zadaná formule musí být ukončena tečkou.

Kvantifikátory se zde zapisují pomocí „klíčových“ písmenek:

- (E x) ... existenční kvantifikátor („existuje takové *x*, pro který platí...“),
- (A x) ... obecný kvantifikátor („pro všechna *x* platí...“).

Navíc kromě výše jmenovaných kvantifikátorů ∀ a ∃ může uživatel zadat i několik dalších, dalo by se říci rozšiřujících:

- (F x) ... „pro nekonečně mnoho *x* platí...“,
- (G x) ... „pro všechna *x*, kterých je konečně mnoho, platí...“,
- (C x) ... „pro souvislou podmnožinu platí...“,
- (Xk x) ... „pro právě *k* hodnot *x* platí...“.

Logické spojky jsou pak reprezentovány speciálními znaky:

a	nebo	negace	implikace	implikace	ekvivalence
`/\`	`\/`	`~`	`==>`	`<==`	`<==>`

Relační operátory se uvádějí tak, jak je uživatel obyčejně zvyklý, za zmínku stojí pouze operátor `/=` odpovídající známému ≠.

Pokud program některému ze zadávaných vstupů nerozumí, nejčastěji kvůli výskytu chyby či nějaké nejednoznačnosti, opakuje žádost o jeho zadání. Jsou-li všechny vstupy zadány v pořádku, může nakonec uživatel nechat proběhnout celý proces eliminace kvantifikátorů pomocí příkazu `finish`, popřípadě postupně projít všechny fáze eliminace příkazem `go`. V takovém případě se uživateli naskýtá možnost použít v každé fázi další z dostupných příkazů programu, které přináší dodatečné informace o právě zpracovávané matematické formuli.

Předveďme postup práce s programem třeba na jednoduché matematické formuli $(\forall x)$: $x^2 \geq 0$. Spustíme program a postupně podle jeho pokynů píšeme zadání:

```
Enter an informal description between ‘[’ and ‘]’:
 [nezapornost mocniny]
```

```
Enter a variable list:
 (x)
Enter the number of free variables:
 0
Enter a prenex formula:
 (A x)[x^2 >= 0].
```

Příkazem `finish` je programu dána instrukce, aby provedl přímo celou eliminaci a vypsal výsledek.

```
An equivalent quantifier-free formula:
 TRUE
```

Výpis odpovědi je následně doplněn informacemi o právě proběhnuvší operaci, včetně času nutného k provedení výpočtu.

Kromě zmíněného příkazu `finish` je možné použít i mnoho dalších příkazů, jejichž seznam uživatel nalezne po zadání příkazu `help`. Jejich pomocí lze kupříkladu krokovat algoritmus eliminace kvantifikátorů nebo zobrazit dodatečné informace k jeho běhu.

Zatím jsme vyzkoušeli eliminaci pouze na jednoduché matematické formuli $(\forall x)$: $x^2 \geq 0$, u níž nám program pouze potvrdil její pravdivost. Zkusme nyní o něco komplikovanější formuli $(\forall x)$: $a \cdot x^2 \geq b$, u které lze předpokládat výstupní ekvivalentní formuli bez kvantifikátorů v zajímavějším tvaru:

```
Enter an informal description between ‘[’ and ‘]’:
 [parametry]
Enter a variable list:
 (a,b,x)
Enter the number of free variables:
 2
Enter a prenex formula:
 (A x)[a x^2 >= b].
```

Odpověď programu je ve tvaru nekvantifikované formule.

```
An equivalent quantifier-free formula:
 a >= 0 /\ b < 0
```

Pokud má tedy platit zadaná formule $(\forall x)$: $a \cdot x^2 \geq b$, musí být proměnná a nezáporná a proměnná b naopak menší než nula.

Přestože QEPCAD nemá samostatný příkaz pro určení času potřebného pro provedení celého procesu eliminace, je i zde vcelku jednoduché tento údaj zjistit. Program jej totiž vypisuje automaticky při svém skončení.

Tedy například při práci s předchozí ilustrační úlohou je uživatel po vypsání výsledku též informován o době přibližně[16] 12 milisekund nutné k eliminování kvantifikátorů.

Samozřejmostí je v QEPCADu i možnost provádění důkazů některých tříd matematických vět. V některých případech je však nutné dokazovanou větu trochu upravit, protože zápis vstupních formulí QEPCADu si neporadí s podíly a odmocninami. Například dokázání věty $(\forall x, y, x \in R^+, y \in R^+)$: $x \cdot y = 1 \Rightarrow x + y \geq \sqrt{2}$ musíme převést na důkaz věty $(\forall x, y, x \in R^+, y \in R^+)$: $x \cdot y = 1 \Rightarrow (x + y)^2 \geq 2$ a ten provedeme následovně[17]:

```
Enter an informal description between '[' and ']':
 [dukaz s implikaci]
Enter a variable list:
 (x,y)
Enter the number of free variables:
 0
Enter a prenex formula:
 (A x)(A y)[[x * y = 1] ==> [(x + y)^2 >= 2]].
```

Po zadání příkazu `finish` k dokončení eliminace program odpoví `TRUE`, dokazovaná věta tedy opravdu platí pro všechna reálná x a y. [BRO02, BRO03]

7.2.3 Příklady v programu QEPCAD B

Jak bylo vidět na několika předvedených příkladech, není v programu QEPCAD jednoduché pracovat s formulemi obsahujícími proměnné, ke kterým se vážou nějaké vstupní podmínky. Tyto podmínky je pro provedení zápisu nutné „opsat“ tím způsobem, že v podstatě zkonstruujeme novou složitější formuli, která však odpovídá té původní s podmínkami.

---

[16] Slovo „přibližně“ zde má své opodstatnění, výsledný strojový čas potřebný pro eliminaci totiž není striktně dán a závisí například i na dalších spuštěných aplikacích, velikosti dostupné operační paměti a podobně.

[17] Díky druhé mocnině součtu $x + y$ v důsledku implikace není nutné zajišťovat jeho kladnou hodnotu, bylo by tedy možné ze zápisu vypustit $x \in R^+$ a $y \in R^+$.

Příklad 7.2.3.1: Vraťme se k optimalizační úloze z příkladu 6.1.3.10. V programu Mathematica jsme použili příkaz **Resolve [ForAll [{x, y}, x ≥ 0 && y ≥ 0 && x + y ≤ 300 && x ≤ 140 && y ≤ 200 && x + y ≥ 90 && x + y ≥ 180, 7x + 8y + 3510 ≥ 7x0 + 8y0 + 3510]]**, jehož argument odpovídá matematické formuli ($\forall\, x, y,\ x \geq 0,\ y \geq 0,\ x + y \leq 300,\ x \leq 140,\ y \leq 200,\ x + y \geq 90,\ x + y \geq 180$): $7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$. Výstupem pak byla formule bez kvantifikátorů `(x0|y0)∈Reals&&−1300+7x0+8y0≤0`, která však o řešení mnoho neříkala, byť jej šlo dopočítat z výsledné nerovnice při zohlednění vstupních podmínek.

V programu QEPCAD není možné zapisovat podmínky pro jednotlivé proměnné přímo do zadané formule jako v programu Mathematica, musíme proto zvolit trochu jiný přístup. Nabízí se nám dohromady dvě možnosti.

Jednou z nich je využití příkazu `assume` spolu s jeho argumentem v podobě nekvantifikované formule obsahující příslušné podmínky, který uživatel napíše hned po zadání vstupní formule. Samotný rozklad cylindrické algebraické dekompozice je poté proveden pouze v oblastech vyhovujících podmínkám.

Druhou možností je vytvoření kvantifikované matematické formule ve tvaru implikace a tyto podmínky zapracovat do jejího předpokladu.

Požijme druhou možnost, máme-li tedy známé podmínky pro proměnné x_0, y_0, x a y, berme je jako předpoklad a jako důsledek pak samotné tvrzení $7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$.

Dostáváme implikaci s několikanásobným předpokladem v podobě konjunkce několika nerovnic (omezující podmínky) implikujícím nerovnici. Tvar formule je ($\forall\, x, y$): $(x \geq 0 \ \wedge\ y \geq 0 \ \wedge\ x + y \leq 300 \wedge\ x \leq 140 \wedge\ y \leq 200 \wedge\ x + y \geq 90 \wedge x + y \geq 180) \Rightarrow (7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510)$.

Do programu QEPCAD pak postupně zadáváme příkazy:

```
Enter an informal description between ‘[’ and ‘]’:
 [optimalizace]
Enter a variable list:
 (x0,y0,x,y)
Enter the number of free variables:
 2
Enter a prenex formula:
```

```
 (A x)(A y)[[x >= 0  /\  y >= 0  /\  x + y <= 300  /\
x <= 140 /\ y <= 200 /\ x + y >= 90 /\ x + y >= 180] ==>
[7 x + 8 y + 3510 >= 7 x0 + 8 y0 + 3510]].
```

Výstupem programu po zadání příkazu `finish` je formule bez kvantifikátorů ve formě `8 y0 + 7 x0 – 1300 <= 0`. Nezískáváme tím sice přímo žádné nové poznatky o hodnotách proměnných x_0, y_0, avšak výsledná nekvantifikovaná formule je shodná s formulí, kterou jsme obdrželi v programu Matematica. jsme tedy na správné stopě.

Dopočtení samotných hodnot uspořádané dvojice $[x_0; y_0]$ by nyní spočívalo v zohlednění podmínek týkajících se těchto proměnných, výsledkem by byl jejich průnik se získanou nerovnicí.

Zkusme je přidat do vstupní formule; celý zápis v QEPCADu by byl obdobný jako výše (včetně seznamu proměnných a počtu volných proměnných) s tím rozdílem, že by tentokrát do řetězce konjunkcí v předpokladu implikace přibyly i omezující podmínky týkající se obou nekvantifikovaných proměnných x_0, y_0.

```
Enter a prenex formula:
 (A x)(A y)[[x0 >= 0  /\  y0 >= 0  /\  x0 + y0 <= 300  /\
x0 <= 140   /\   y0 <= 200   /\   x0 + y0 >= 90   /\
x0 + y0 >= 180  /\  x >= 0  /\  y >= 0  /\  x + y <= 300  /\
x <= 140 /\ y <= 200 /\ x + y >= 90 /\ x + y >= 180] ==>
[7 x + 8 y + 3510 >= 7 x0 + 8 y0 + 3510]].
```

Výsledek nyní dostaneme ve formě řetězce disjunkcí, který si zaslouží bližší prozkoumání, abychom jej mohli správně interpretovat:

```
y0 + x0 – 300 > 0   \/   x0 < 0   \/   y0 – 200 > 0   \/
x0 – 140 > 0  \/  y0 + x0 – 180 < 0  \/  [x0 – 140 = 0  /\
y0 + x0 – 180 = 0]
```

Prvních pět nerovnic řetězce disjunkcí se může na první pohled zdát matoucích, nicméně uživatel si musí uvědomit, že zadaná formule je implikací a ta je pravdivá i tehdy, není-li splněn její předpoklad. Tomuto nepravdivému předpokladu odpovídají právě tyto nerovnice $x_0 + y_0 > 300$, $x_0 < 0$, atd.

Z našeho pohledu nejdůležitější je až poslední část odpovědi mající tvar konjunkce $x_0 = 140 \wedge x_0 + y_0 = 180$. Odtud již dostaneme správný výsledek $x_0 = 140$ a $y_0 = 40$, jehož jsme dosáhli trochu jinou cestou i v programu Mathematica.♦

Příklad 7.2.3.2: Nalezněte řešení soustavy rovnic

$$x + y - z = 7$$
$$x^2 + y^2 - z^2 = 37$$
$$x^3 + y^3 - z^3 = 1.$$

Při „lidském“ řešení by v této úloze bylo zřejmě nejvhodnější nejprve z první rovnice vyjádřit součet $x + y = z + 7$, dosadit jej do druhé rovnice, kde bychom následně dostali rovnost $xy = 7z + 6$, a nakonec obě rovnosti dosadit do rovnice třetí, odkud bychom již obdrželi neznámou z.

Pokud budeme chtít nalézt řešení prostřednictvím programu QEPCAD, musíme opět nejprve sestavit příslušnou formuli. Předpokládejme, že existuje nějaké reálné x, jenž je řešením. Tuto skutečnost můžeme přepsat do formule $(\exists x)$: $x + y - z = 7 \wedge x^2 + y^2 - z^2 = 37 \wedge x^3 + y^3 - z^3 = 1$. Do příkazového řádku programu QEPCAD potom zadáváme posloupnost příkazů:

```
Enter an informal description between ‘[’ and ‘]’:
 [soustava]
Enter a variable list:
 (y,z,x)
Enter the number of free variables:
 2
Enter a prenex formula:
 (E x)[x + y – z = 7 /\ x^2 + y^2 – z^2 = 37 /\ x^3 + y^
3 – z^3 = 1].
```

Po skončení eliminace nám program odpoví výpisem formule bez kvantifikátorů

```
y – 9 >= 0 /\ y – 10 <= 0 /\ y z – 7 z – y^2 + 7 y – 6 =
0 /\ [y – 10 = 0 \/ y – 9 = 0].
```

První dvě nerovnice výstupu říkají, že hledaná hodnota neznámé y leží v intervalu $\langle 9; 10\rangle$, zároveň poslední část formule `[y – 10 = 0 \/ y – 9 = 0]` již přímo určuje dvě hodnoty $y = 9$ a $y = 10$, které jsou řešením. Poslední rovnici obsahující i neznámou z pak použijeme k jejímu dopočítání[18]. Určení poslední

[18] Zde lze bez problémů použít formuli $(\exists y)$: $(-y^2 + 7y - 7z + yz - 6 = 0) \wedge (y = 9 \vee y = 10)$ a provést eliminaci kvantifikátorů.

neznámé x je pak již triviální záležitostí. Nalezli jsme dvě řešení, kterými jsou uspořádané trojice $[x, y, z]$ o hodnotách [10, 9, 12] a [9, 10, 12]. ♦

7.3 Maple

7.3.1 Program Maple

Jak již bylo zmíněno výše, program Maple neprovádí eliminaci kvantifikátorů jako takovou. Prostřednictvím trojúhelníkové dekompozice dokáže řešit soustavy polynomiálních rovnic a nerovnic. Pokud k některé z vyskytnuvších se proměnných přistupujeme jako k parametru, Maple najde a vypíše příslušná řešení právě ve vztahu k tomuto parametru. Jde sice o odlišný výstup, než jaký poskytuje samotná eliminace kvantifikátorů prováděná pomocí cylindrické algebraické dekompozice, avšak při správném způsobu interpretace je z něj možné obdržet matematickou formuli bez kvantifikátorů, která je ekvivalentní s formulí, jenž odpovídá zadané polynomiální soustavě. Navíc Maple má příkazy také pro provedení samotného rozkladu na jednotlivé buňky a taktéž vypsání zjištěných testovacích bodů.

Z tohoto důvodu je nutné zahrnout Maple mezi programy počítačové algebry, kterým věnujeme pozornost.

Maple je, obdobně jako Wolfram Mathematica, komerčním programem počítačové algebry. Jeho první koncept byl navržen již roku 1980 na Waterlooské univerzitě v Ontariu v Kanadě. Cílem bylo vytvořit počítačový algebraický systém, jenž by neměl velké výkonnostní nároky na počítačové sestavy, na nichž je spuštěn.

V současné době jej dále vyvíjí společnost Waterloo Maple Inc., na trh jej však uvádí pod jménem Maplesoft.[19] [MAP12]

7.3.2 Příkazy programu Maple

Stejně jako v programu Mathematica existují i v programu Maple pro práci s kvantifikátory a k zápisu matematických formulí příslušné příkazy. Jsou jimi příkazy *forall* (*x*, *tvrzení*) (případně *forall* ([*x*, *y*], *tvrzení*), chceme-li vyjádřit, že v zápisu pracujeme s uspořádanou dvojicí $[x, y]$, neboť podobný zápis ve tvaru *forall* (*x*, *y*, *tvrzení*) je programem chápán jako zkrácená varianta příkazu *forall* (*x*, *forall* (*y*, *tvrzení*))) a *exists* (*x*, *tvrzení*) (obdobným přístupem lze pracovat i se zápisem

[19] Jméno samotného počítačového programu Maple i jméno distribuční společnosti Maplesoft má odkazovat na zemi původu (maple = javor).

exists ([*x*, *y*], *tvrzení*) pro uspořádanou dvojici proměnných a *exists* (*x*, *y*, *tvrzení*) pro zkrácený zápis *exists* (*x*, *exists* (*y*, *tvrzení*))).

Například pro zapsání kvantifikované matematické formule $(\forall x)(\exists y)$: $x^3 - 2y + 5 > 0$ tedy použijeme příkaz *forall* (*x*, *exists* (*y*, $x^3 - 2\,y + 5 > 0$)), na což program reaguje vypsáním odpovídajícího zápisu *forall*(*x*, *exists*(*y*, $0 < x^3 - 2\,y + 5$)). Pokud uživatel chce zadat formuli obsahující soustavu více rovnic nebo nerovnic, musí opět použít logické spojky, které jsou však v programu Maple oproti Mathematice ve formě **and** a **or**. Zápis takové formule pak může vypadat následovně: *exists* (*x*, *exists* (*y*, $x \cdot y > 1$ **and** $x^2 + y = 0$)). Program její úspěšné zadání potvrdí výpisem *exists*(*x*, *exists*(*y*, $1 < xy$ **and** $x^2 + y = 0$)).

Zaměřme nejprve pozornost na samotnou cylindrickou algebraickou dekompozici, pro kterou v programu Maple existuje příkaz *CylindricalAlgebraicDecompose* (*M*, *okruh*), kde jako první argument příkazu *M* vystupuje množina daných polynomů a druhým argumentem *okruh* je okruh polynomů. Takto zadaný příkaz pak uživateli vrátí *M*-invariantní cylindrickou algebraickou dekompozici příslušného *n*-rozměrného prostoru, a to v podobě seznamu jednotlivých buněk rozkladu.

Zkusme určit rozklad pro polynom $p(x, y) = x \cdot y - 1$. Nejprve je nutné programu určit, se kterými balíčky příkazů budeme pracovat, pro výše zmíněný příkaz jimi jsou balíčky *RegularChains*, *ChainTools* a *SemiAlgebraicTools*. Bez nich není možné operaci provést.

Zavoláme je tedy příkazy *with*(*RegularChains*), *with*(*ChainTools*) a *with*(*SemiAlgebraicSetTools*). Na všechny počítač reaguje vypsáním potvrzující zprávy a seznamu příkazů volaného balíčku.

Nyní již můžeme psát samotný příkaz *CylindricalAlgebraicDecompose* ([$x \cdot y - 1$], *PolynomialRing* ([*x*, *y*])), na nějž program odpoví výpisem všech buněk rozkladu, a to v následujícím kompletním tvaru[20]:

[*regular_chain*, [[−1, −1], [−2, −2]]] $\quad x < \frac{1}{y}$

[*regular_chain*, [[−1, −1], [−1, −1]]] $\quad x = \frac{1}{y} \quad y < 0$

[20] Volitelným argumentem příkazu *CylindricalAlgebraicDecompose* je také možnost výběru, jakým způsobem se má výstup zobrazit. Lze tak učinit pomocí tabulky, stromu nebo seznamu, přičemž tabulka bývá zpravidla nejpřehlednější.

[*regular_chain*, [[−1, −1], [0, 0]]] $\frac{1}{y} < x$

[*regular_chain*, [[0, 0], [0, 0]]] $y = 0$

[*regular_chain*, [[1, 1], [0, 0]]] $x < \frac{1}{y}$

[*regular_chain*, [[1, 1], [1, 1]]] $x = \frac{1}{y}$ $y > 0$

[*regular_chain*, [[1, 1], [2, 2]]] $\frac{1}{y} < x$

Zobrazený výstup popisuje jednotlivé buňky, kterých je v tomto případě sedm. Pro proměnnou y platí rozdělení na případy $y < 0$, $y = 0$, $y > 0$ (označme tyto případy D_1, D_2 a D_3), přičemž v prvním a posledním případě jsou tyto oblasti (sektory) rozděleny podle hodnoty proměnné x ještě na další tři samostatné části (D_{11}, D_{12}, D_{13} a D_{31}, D_{32}, D_{33}).

Poznámka: Příkazy je do sebe možné vnořovat, jak jsme již předvedli v předchozím příkladě, nebo využít přiřazování do zavedených proměnných. V takovém případě bychom psali:

M := [*x*·*y* − 1]

okruh := *PolynomialRing* ([*x*, *y*])

CylindricalAlgebraicDecompose (*M*, *okruh*),

výstup programu by však byl shodný. Pro přehlednost v zapsaných příkazech se nadále budeme držet tohoto přístupu. ♦

Se zvyšující se složitostí zadaných křivek samozřejmě stoupá i složitost zjištěného a vypsaného rozkladu, jako příklad můžeme udat na první pohled poměrně jednoduchou úlohu, kdy prvním argumentem příkazu bude polynom $x^2 + y^2 - 1$, jehož zobrazením v rovině je jednotková kružnice.

Píšeme:

M := [$x^2 + y^2 - 1$]

okruh := P*olynomialRing* ([*x*, *y*])

CylindricalAlgebraicDecompose (*M*, *okruh*)),

na což program odpoví výpisem buněk rozkladu. Uvedeme zde pouze interpretaci popisu jednotlivých buněk rozkladu pomocí příslušných rovností a nerovností (kompletní výpis výstupu tohoto příkazu je v Příloze 3):

$y < -1$

	$x < 0$
$y = -1$	$x = 0$
	$x > 0$
	$x < -\sqrt{1-y^2}$
	$x = -\sqrt{1-y^2}$
$-1 < y < 1$	$-\sqrt{1-y^2} < x < \sqrt{1-y^2}$
	$x = \sqrt{1-y^2}$
	$x > \sqrt{1-y^2}$
	$x < 0$
$y = 1$	$x = 0$
	$x > 0$
$y > 1$	

Podle prvního sloupce pro proměnnou *y* dostáváme 3 sektory a 2 sekce (značme je popořadě D_1, D_2, ..., D_5), přičemž dvě sekce a jeden sektor jsou pak dále ještě rozděleny (D_{21}, D_{22}, D_{23}, D_{31}, ..., D_{43}). Celkově tak dostáváme dohromady 13 buněk rozkladu.

Povšimněme si, že oproti programu Wolfram Mathematica, kde můžeme využít obdobný příkaz **CylindricalDecomposition**, nejsme v případě programu Maple striktně omezeni tvarem vstupu příkazu. V Mathematice bylo nutné použít jako argument příslušného příkazu rovnost nebo nerovnost, pro získání kompletního rozkladu celého *n*-rozměrného prostoru jsme tedy museli postupně zadat rovnost i obě nerovnosti s daným výrazem, podle něhož dekompozici provádíme[21].

V programu Maple je ale argumentem příkazu *CylindricalAlgebraicDecompose* pouhý výraz, který je položen rovný nule a rovnou je podle této rovnosti provedena kompletní dekompozice celého prostoru.

Nezůstaneme ale jen u pouhého rozkladu, program Maple má také příkaz pro určení vzorku testovacích bodů, a tím je *SamplePoints* (*soustava*, *okruh*), kde argument *soustava* je zadaná soustava rovnic a nerovnic a *okruh* je opět okruh polynomů. Nyní je už nutné pracovat přímo s rovnicemi a nerovnicemi, nestačí se zabývat jen příslušnými

[21] Toto navíc platilo v případě, že jsme pracovali pouze s jednou rovnicí nebo nerovnicí. V situaci, kdy pracujeme se soustavou rovnic a nerovnic, bychom pak mohli být nuceni zadat všechny možné kombinace relačních operátorů „=“, „<“ a „>“.

výrazy. Určení testovacích bodů je provedeno příkazem *SamplePoints* (*soustava, okruh*), výpis vzorku pak příkazem *Display* (*SamplePoints* (*soustava*, *okruh*), *okruh*). Zkusme programu zadat úkol nalézt testovací body pro rozklady provedené v předchozích ukázkách.

Pro získání testovacích bodů rozkladu podle výrazu $x \cdot y - 1$ postupně píšeme příkazy

soustava := [$x \cdot y - 1 = 0$]

okruh := *PolynomialRing* ([*x*, *y*]

vzorek := *SamplePoints* (*soustava, okruh*)

Display (*vzorek*, *okruh*)

na něž obdržíme výstup ve tvaru:

$$x = -2 \qquad x = 2$$

$$y = -\frac{1}{2} \qquad y = \frac{1}{2}$$

Obdobným způsobem provedeme výpočet i pro nerovnost $x \cdot y - 1 \neq 0$, výstupem jsou dvojice souřadnic bodů:

$$x = -\frac{9}{2} \qquad x = \frac{1}{2} \qquad x = -\frac{1}{2} \qquad x = \frac{9}{2}$$

$$y = -\frac{1}{2} \qquad y = -\frac{1}{2} \qquad y = \frac{1}{2} \qquad y = \frac{1}{2}$$

Kompletní množina vzorku testovacích bodů je pak

$$s = \left(\left[-\frac{9}{2}; -\frac{1}{2}\right], \left[-2; -\frac{1}{2}\right], \left[-\frac{1}{2}; \frac{1}{2}\right], \left[\frac{1}{2}; -\frac{1}{2}\right], \left[2; \frac{1}{2}\right], \left[\frac{9}{2}; \frac{1}{2}\right]\right).$$

Provedeme-li zjištění testovacích bodů i pro druhý příklad s polynomem $x^2 + y^2 - 1$ pomocí příkazů

soustava := [$x^2 + y^2 - 1 = 0$], resp. *soustava* := [$x^2 + y^2 - 1 \neq 0$]

okruh := *PolynomialRing* ([*x, y*])

vzorek := *SamplePoints* (*soustava, okruh*)

Display (*vzorek*, *okruh*),

dostaneme v prvním případě ($x^2 + y^2 - 1 = 0$) souřadnice testovacích bodů:

$$x = -1 \qquad x = 1 \qquad x = 0 \qquad x = 0$$

$$y = 0 \qquad y = 0 \qquad y = 1 \qquad y = -1$$

Ve druhém případě ($x^2 + y^2 - 1 \neq 0$) jsou výstupem souřadnice:

$$x = 0 \qquad x = -\frac{3}{2} \qquad x = 0 \qquad x = \frac{3}{2} \qquad x = 0$$

$$y = -\frac{3}{2} \qquad y = 0 \qquad y = 0 \qquad y = 0 \qquad y = \frac{3}{2}$$

Kompletní vzorek testovacích bodů je

$$s = \left(\left[-\frac{3}{2}; 0\right], [-1; 0], \left[0; -\frac{3}{2}\right], [0; -1], [0; 0], [0; 1], \left[0; -\frac{3}{2}\right], [1; 0], \left[\frac{3}{2}; 0\right]\right).$$

[MAP12]

7.3.3 Příklady v programu Maple

Jak již bylo napsáno výše, provedení eliminace kvantifikátorů není v programu Maple tak jednoduchou záležitostí jako v programech Mathematica a QEPCAD, neexistuje pro ni ani samotný příkaz. Avšak jistých výsledků lze přesto dosáhnout, a to v případech, že budeme ke kvantifikovaným formulím přistupovat z odlišného úhlu.

Zadanou kvantifikovanou formuli můžeme vnímat též jako soustavu rovnic a nerovnic, v nichž s nekvantifikovanými proměnnými zacházíme jako s parametry. Řešení problému v takovém případě přechází v otázku, za jakých podmínek, přesněji pro jaké hodnoty těchto parametrů, má soustava řešení. Takto získaný výstup je pak možné interpretovat také jako hledanou formuli bez kvantifikátorů.

Celý postup s sebou ale přináší nemalé komplikace. Jeho zápis v programu Maple je relativně složitý, zadávaná soustava rovnic a nerovnic musí být přísně polynomiálního charakteru a v neposlední řadě je po uživateli při zadávání úlohy do programu požadován počet řešení, který pro daný parametr soustava má mít.

Přesto si ukažme několik jednoduchých úloh a práci s nimi.

Příklad 7.3.3.1: Proveďte eliminaci kvantifikátorů z kvantifikované formule $(\exists x)$: $a \cdot x^2 + b \cdot x + c = 0$.

Nahlížejme na zadanou formuli jako na polynomiální rovnici $a \cdot x^2 + b \cdot x + c = 0$, přičemž x je proměnná a a, b a c chápeme jako parametry, nikoliv jako nekvantifikované proměnné. Úkolem nyní bude určit hodnoty těchto parametrů tak, aby rovnice měla řešení.

K tomu využijeme balíčky *RegularChains*, *ParametricSystemTools* a *SemiAlgebraicSetTools* a následně příkazy *RealRootClassification*, *RepresentingBox* a *RepresentingQuantifierFreeFormula*.

Začneme přiřazením:

okruh := *PolynomialRing* ([*x*, *a*, *b*, *c*])

rovnice := [$a \cdot x^2 + b \cdot x + c$]

nezapornevyrazy := [ ]

kladnevyrazy := [ ]

nerovnice := [ ],

kde *rovnice* je množina výrazů rovných nule, *nezapornevyrazy* je prázdná množina nezáporných výrazů, *kladnevyrazy* množina kladných výrazů a *nerovnice* množina polynomiálních nerovnic.

Uplatníme příkaz

rrc := *RealRootClassification* (*rovnice*, *nezapornevyrazy*, *kladnevyrazy*, *nerovnice*, 3, 2, *okruh*),

kde pátý argument – číslo 3 – určuje, že poslední tři proměnné z daného okruhu jsou považovány za parametry, a šestý argument – číslo 2 – udává počet řešení, která má soustava mít.

Jinými slovy hledáme hodnoty parametrů *a*, *b* a *c*, pro něž má kvadratická rovnice $a \cdot x^2 + b \cdot x + c = 0$ dva různé kořeny.

Hledání řešení dokončíme zapsáním příkazů

rb := *RepresentingBox* (*rrc* [1, 1], *okruh*)

qff := *RepresentingQuantifierFreeFormula* (*rb*)

DisplayQuantifierFreeFormula (*qff*).

Výstupem v tomto případě bude podmínka pro parametry, kterou můžeme chápat jako formuli bez kvantifikátorů ve tvaru $4ac - b^2 < 0$. Neboli pokud platí, že diskriminant $D = b^2 - 4ac$ je kladný, má zadaná parametrická kvadratická rovnice dva různé kořeny. ♦

Stejným způsobem můžeme ověřit i to, za jakých podmínek rovnice řešení mít nebude, stačí v příkazu *RealRootClassification* položit šestý argument rovný 0. V takovém případě dostaneme výslednou odpověď ve tvaru $4ac - b^2 > 0$.

Problém nastane ve chvíli, kdy bychom chtěli znát hodnoty parametrů pro případ, kdy existuje pouze jeden dvojnásobný kořen. Tentokrát nemá funkce *RealRootClassification* validní výstup a řešení celého problému neobdržíme.

Podobně problematické může být hledání podmínek pro parametry i v dalších podobných úlohách.

Příklad 7.3.3.2: Určete hodnotu parametru *a*, pro niž existuje řešení soustavy rovnic

$$x^2 = y^2$$

$$(x - a)^2 + y^2 = 1.$$

Kvantifikovaná formule pro tuto úlohu je ve tvaru $(\exists x, y)$: $x^2 = y^2 \wedge (x - a)^2 + y^2 = 1$.

Pro řešení problému postupně zapisujeme příkazy

okruh := *PolynomialRing* ([*x*, *y*, *a*])

rovnice := [$x^2 - y^2$, $(x - a)^2 + y^2$]

rrc := *RealRootClasssification* (*rovnice*, [ ], [ ], [ ], 1, 4, *okruh*)

rb := *RepresentingBox* (*rrc* [1, 1], *okruh*)

qff := *RepresentingQuantifierFreeFormula* (*rb*)

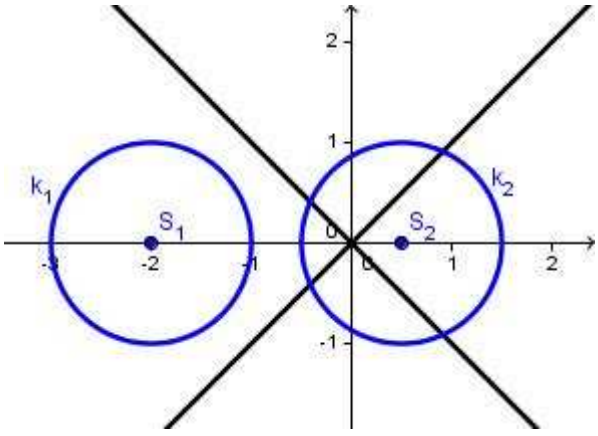
DisplayQuantifierFreeFormula

Hledaným vyjádřením podmínek pro parametr *a* je formule bez kvantifikátorů $a^2 < 2$, čili pro hodnoty *a* spadající do intervalu $(-\sqrt{2}\,;\sqrt{2}\,)$ by měla existovat čtyři řešení.

Naopak pro zadání *rrc* := *RealRootClassification* (*rovnice*, [ ], [ ], [ ], 1, 0, *okruh*) , kde šestý argument s hodnotou 0 znamená, že hledáme podmínky, při nichž soustava nemá řešení, program odpoví formulí $a^2 - 2 > 0$, hodnota *a* je tentokrát ze sjednocení intervalů $(-\infty; -\sqrt{2}\,) \cup (\sqrt{2}\,; \infty)$.

Podívejme se ale na celý problém z jiného úhlu, a sice pohledem analytické geometrie. Rovnice $x^2 = y^2$ odpovídá osám kvadrantů, druhá rovnice $(x - a)^2 + y^2 = 1$ reprezentuje jednotkovou kružnici se středem v bodě $S = [a; 0]$. V závislosti na hodnotě parametru *a* se pak mění pozice středu a celé kružnice, čemuž ve výsledku odpovídá i počet průsečíků os kvadrantů s kružnicí, jinak též řešení soustavy.

Jak je patrné z obrázku, jednotlivých případů s různými počty řešení, bude více než pouze dva výše popsané.



Obrázek 25: Dvě z možných poloh kružnice $(x-a)^2 + y = 1$ s různými počty průsečíků s osami kvadrantů

Celou situaci lze přehledně shrnout do tabulky:

	počet řešení	
a	ve skutečnosti	podle Maple
$(-\infty; -\sqrt{2}) \cup (\sqrt{2}; \infty)$	0	0
$\{\pm\sqrt{2}\}$	2	-
$(-\sqrt{2}; -1) \cup (1; \sqrt{2})$	4	4
$\{\pm 1\}$	3	4
$(-1; 1)$	4	4

Z výše uvedeného je zřejmé, že tento přístup k eliminaci kvantifikátorů prostřednictvím programu Maple není ten nejšťastnější, byť v určitých jednodušších případech fungovat může.♦

Nakonec ještě zmiňme, že obdobně jako v programu Mathematica existuje i v programu Maple příkaz ke zjištění doby výpočtu zadaného problému, a sice příkaz *time* (*výraz*).

7.4 Malé srovnání

Na závěr proveďme malé experimentální srovnání.

Hodnotit a porovnávat „jednoduchost“ získaných výstupních formulí by mohlo být u takto rozdílných programů poměrně složité, proto se raději zaměřme na porovnání času potřebného k samotnému provedení eliminace jednotlivými programy v různých zadaných formulích.

Pro tento účel vyberme z kapitol 7.1, 7.2 a 7.3 několik již prezentovaných úloh, resp. matematických formulí s kvantifikovanými proměnnými, které postupně počítači zadáme ke zjednodušení ve všech třech programech. Každý výpočet pětkrát zopakujeme, abychom se vyvarovali náhodných výkyvů. Kromě průměrné doby běhu procesu budeme zapisovat i nejdelší a nejkratší čas potřebný k provedení operace.

7.4.1 Soubor testovacích úloh

1) rozklad cylindrickou algebraickou dekompozicí:

   rozklad pro nerovnost $x^2 + y^2 - 1 \leq 0$

2) určení testovacích bodů rozkladu:

   testovací body pro nerovnost $16y^2 + 4x^3 - 4x \leq 1$

3) důkaz matematické věty s implikací:

$$(\forall x, y, x \in R^+, y \in R^+): x \cdot y = 1 \Rightarrow x + y \geq \sqrt{2}$$

zadání pro program Mathematica ve tvaru $(\forall x, y, x \in R^+, y \in R^+, x \cdot y = 1)$: $x + y \geq \sqrt{2}$,

zadání pro program QEPCAD ve tvaru $(\forall x, y)$: $x > 0 \wedge y > 0 \wedge x \cdot y = 1 \Rightarrow (x + y)^2 \geq 2$,

4) limita funkce:

funkce $f(x, y) = \dfrac{5x}{3x - 2}$

zadání pro program Mathematica ve tvaru $(\forall \varepsilon, \varepsilon > 0)(\exists x_0)(\forall x, x > x_0)$: $a - \varepsilon < \dfrac{5x}{3x - 2} < a + \varepsilon$,

zadání pro program QEPCAD ve tvaru $(\forall \varepsilon)(\exists x_0)(\forall x)$: $(\varepsilon > 0 \wedge x > x_0) \Rightarrow (3ax - 3\varepsilon x - 2a + 2\varepsilon < 5x \wedge 5x < 3ax + 3\varepsilon x - 2a - 2\varepsilon)$,

5) omezenost funkce:

funkce $f(x, y) = 2x^2 + y^2 - 2y + 5$

zadání pro program Mathematica ve tvaru $(\exists K, K > 0)(\forall x)$: $-K < 2x^2 + y^2 - 2y + 5 < K$,

zadání pro program QEPCAD ve tvaru $(\exists K)(\forall x)(\forall y)$: $-K < 2x^2 + y^2 - 2y + 5 \wedge 2x^2 + y^2 - 2y + 5 < K$,

6) průsečík křivek:

křivky $x^2 + y^2 = 4$, $y^2 = x + 1$

zadání pro program Mathematica ve tvaru $(\exists x, y)$: $x^2 + y^2 - 4 = 0 \wedge x - y^2 + 1 = 0$,

zadání pro program QEPCAD ve tvaru $(\exists x)(\exists y)$: $x^2 + y^2 - 4 = 0 \wedge x - y^2 + 1 = 0$,

zadání pro program Maple ve tvaru soustavy, jedna proměnná je brána jako parametr:

$$x^2 + y^2 - 4 = 0$$

$$x - y^2 + 1 = 0$$

7) soustava 2 rovnic o dvou neznámých s parametrem:

zadání pro program Mathematica ve tvaru $(\exists x, y)$: $x^2 = y^2 \wedge (x - a)^2 + y^2 = 1$,

zadání pro program QEPCAD ve tvaru $(\exists x)(\exists y)$: $x^2 = y^2 \wedge (x - a)^2 + y^2 = 1$

zadání pro program Maple ve tvaru soustavy dvou rovnic o dvou neznámých s jedním parametrem:

$$x^2 = y^2$$

$$(x - a)^2 + y^2 = 1$$

8) soustava 3 rovnic o třech neznámých:

$$x + y - z = 7$$

$$x^2 + y^2 - z^2 = 37$$

$$x^3 + y^3 - z^3 = 1$$

zadání pro program Mathematica ve tvaru $(\exists x)$: $x + y - z = 7 \wedge x^2 + y^2 - z^2 = 37 \wedge x^3 + y^3 - z^3 = 1$,

zadání pro program QEPCAD ve tvaru $(\exists x)$: $x + y - z = 7 \wedge x^2 + y^2 - z^2 = 37 \wedge x^3 + y^3 - z^3 = 1$,

9) slovní úloha:

jednoduchá optimalizační úloha z příkladu 7.1.3.10

zadání pro program Mathematica ve tvaru $(\exists x_0,\ x_0 \geq 0,\ y_0 \geq 0,\ x_0 + y_0 \leq 300,\ x_0 \leq 140,\ y_0 \leq 200,\ x_0 + y_0 \geq 180)(\forall\, x, y,\ x \geq 0,\ y \geq 0,\ x + y \leq 300,\ x \leq 140,\ y \leq 200,\ x + y \geq 180)$: $7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$,

zadání pro program QEPCAD ve tvaru $(\exists x_0)(\forall x, y)$: $(x_0 \geq 0 \wedge y_0 \geq 0 \wedge x_0 + y_0 \leq 300 \wedge x_0 \leq 140 \wedge y_0 \leq 200 \wedge x_0 + y_0 \geq 180 \wedge x \geq 0 \wedge y \geq 0 \wedge x + y \leq 300 \wedge x \leq 140 \wedge y \leq 200 \wedge x + y \geq 180) \Rightarrow 7x + 8y + 3510 \geq 7x_0 + 8y_0 + 3510$,

7.4.2 Tabulky výsledků

Wolfram Mathematica

	pokus							
úloha	1.	2.	3.	4.	5.	min	∅	max
1)	0,015	0	0	0,016	0,015	0	0,009	0,016
2)	0	0	0,015	0,015	0,015	0	0,009	0,015
3)	0	0	0	0,015	0	0	0,003	0,015
4)	0,109	0,110	0,110	0,109	0,109	0,109	0,109	0,110
5)	0,015	0,016	0,016	0,015	0,016	0,015	0,016	0,016
6)	0	0	0,015	0	0	0	0,003	0,015
7)	0	0,016	0,016	0,015	0	0	0,009	0,016
8)	0,016	0,016	0,015	0,016	0,016	0,015	0,016	0,016
9)	0,468	0,485	0,484	0,469	0,485	0,468	0,478	0,485

Poznámka: Nulové časové hodnoty neznamenají, že si výpočet nevyžádal žádný čas, ve skutečnosti byl tento časový úsek tak krátký, že jej program vypsal ve tvaru `0.`, čímž je uživateli dáno najevo, že hodnota je malá, nikoliv však nulová.♦

QEPCAD

	pokus							
úloha	1.	2.	3.	4.	5.	min	∅	max
1)	-	-	-	-	-	-	-	-
2)	-	-	-	-	-	-	-	-
3)	0,020	0,023	0,024	0,022	0,020	0,020	0,022	0,024
4)	0,028	0,032	0,031	0,033	0,030	0,028	0,031	0,033
5)	0,029	0,027	0,027	0,024	0,027	0,024	0,027	0,029
6)	0,020	0,012	0,017	0,021	0,018	0,012	0,018	0,021
7)	0,025	0,026	0,021	0,024	0,023	0,021	0,24	0,026
8)	0,347	0,343	0,350	0,346	0,343	0,343	0,346	0,350
9)	0,237	0,228	0,240	0,231	0,227	0,227	0,233	0,240

Maple

	pokus							
úloha	1.	2.	3.	4.	5.	min	∅	max
1)	0,031	0,031	0,015	0,046	0,031	0,015	0,031	0,046
2)	0,015	0,062	0,015	0,031	0,015	0,015	0,028	0,062
3)	-	-	-	-	-	-	-	-
4)	-	-	-	-	-	-	-	-
5)	-	-	-	-	-	-	-	-
6)	0,032	0,031	0,015	0,016	0,031	0,015	0,025	0,032
7)	0,328	0,344	0,335	0,324	0,340	0,328	0,334	0,344
8)	-	-	-	-	-	-	-	-
9)	-	-	-	-	-	-	-	-

Poznámka: Situace, kdy musí být v programu Maple jednotlivé matematické formule zapisovány prostřednictvím soustav rovnic a nerovnic s parametrem, je dosti složitá. Vzhledem k tomu je řešení zadaných úloh problematické, nehodí se pro všechny z nich a ne vždy je výstup v pořádku (jak je zřejmé kupříkladu v řešení úlohy 7.3.3.2).♦

7.4.3 Porovnání výsledků

Z provedeného porovnání je možné si odnést několik různých závěrů.

Především je nutné si uvědomit, že jakkoliv se uživatel může snažit přizpůsobit zadaný problém vstupním požadavkům programu Maple, ne vždy uspěje. V nemalé míře totiž naráží na úskalí při „převádění“ problému popsaného kvantifikovanou formulí na problém spočívající v hledání řešení soustavy rovnic či nerovnic v závislosti na obsažených parametrech.

Na druhé straně programy Mathematica i QEPCAD si vedly obstojně, dokázaly uspokojivě zvládnout všechny úlohy zaměřené na eliminaci kvantifikátorů. Bráno čistě z pohledu strojového času nutného pro řešení se jeví jako rychlejší program Mathematica, který ve všech srovnávaných příkladech dosáhl rychlejšího zpracování. Pokud bychom však zohledňovali i jednotlivé výstupní formule, ukázal se program QEPCAD i za cenu delšího potřebného času jako zdatnější prostředek, protože některé výstupní formule dokázal vyjádřit v méně komplikovaném tvaru, konkrétně třeba v případě testovací úlohy 7) soustavy dvou rovnic o dvou neznámých s parametrem

$$x^2 = y^2$$

$$(x - a)^2 + y^2 = 1$$

dokázal program Mathematica poskytnout pouhé konstatování `True`, tedy že skutečně existuje alespoň jedna dvojice neznámých x a y, pro něž daná soustava platí. Naproti tomu aplikace QEPCAD uživateli zobrazila výstupní formuli bez kvantifikátorů ve tvaru `−2 + a^2 < 0`, kterážto odpovídá tomu, jak by v daném případě měl výstup vypadat.

8 Závěr

V této práci jsme se zabývali eliminací kvantifikátorů, která i přesto, že je poměrně mladou metodou umožňující řešit dosti široké spektrum matematických úloh, urazila i za krátký čas několika desetiletí dlouhou cestu ve svém vývoji od nejstarších způsobů určování existence kořenů polynomů v jedné proměnné, či zjišťování jejich počtu, přes poněkud neohrabanou a vcelku komplikovanou Tarského eliminační metodu z první poloviny 20. století, až po vcelku rychlé a nejrůznějšími způsoby stále ještě optimalizované přístupy využívající k řešení prostředky výpočetní techniky ve spojitosti s cylindrickou algebraickou dekompozicí nebo nověji trojúhelníkovou dekompozicí.

Z původně složité a můžeme říci i nepříliš perspektivní myšlenky, jejíž komplikovanost a časová náročnost převažovaly nad jejím praktickým uplatněním, se ale časem – a za výrazného přispění rozvoje výpočetní techniky – stala užitečná a výkonná metoda pro řešení širokého spektra matematických úloh od dokazování některých tříd matematických vět, přes řešení rovnic a nerovnic a jejich soustav, až například po vyšetřování parametrických systémů funkcí.

Jak jsme předvedli v kapitole věnované základu cylindrické algebraické dekompozice, je tato metoda vcelku elegantní, jednoduchá a dostupná pro pochopení každému uživateli, částečně i proto, že je možné celou situaci v prostorech nižších dimenzí graficky znázornit.

Zajímavé se z tohoto pohledu jeví i srovnání lidského a počítačového přístupu. Zatímco lidský přístup je v případě práce v jednorozměrném či dvourozměrném prostoru R či R^2 ještě vcelku schůdný, pro prostory vyšších dimenzí to již neplatí a lidský přístup nutně selhává. Na druhé straně algoritmizovaný řešitelský přístup, jejž využívají počítačové programy, sice obsahuje velké množství počítání, které nemusí být pro každého atraktivní, avšak právě díky tomu, že nemusí spoléhat pouze na grafickou interpretaci situace, nejsou v podstatě nikterak limitovány dimenzí prostoru, respektive počtem proměnných, s nimiž pracují.

V současné době patří mezi prostředky, s jejichž pomocí jde eliminaci kvantifikátorů strojově provádět, několik programů počítačové algebry, jmenovitě program Mathematica od společnosti Wolfram Research dostupný nejen pro operační systémy MS Windows a dále také aplikaci QEPCAD B pracující pod operačním systémem Linux a určenou přímo pro nacházení ekvivalentních nekvantifikovaných formulí prostřednictvím částečné cylindrické algebraické dekompozice.

Částečně pak lze také využít program počítačové algebry Maple od společnosti Maplesoft, který však není k provádění eliminace kvantifikátorů vždy úplně vhodný, neboť při práci s ním musí být celá problematika pojata nikoliv jako odstranění kvantifikátorů z matematických formulí užitím cylindrické algebraické dekompozice, ale jako řešení soustav rovnic či nerovnic s parametry použitím trojúhelníkové dekompozice. Z tohoto pak plynou i některé problémy, které se mohou při řešení zadaných úloh objevit, jak bylo ukázáno na konci kapitoly 7.3.3. Právě pro tento důvod není v kapitolách 7.4.1 a 7.4.2 řešena programem Maple kompletní sada testovacích úloh.

Používání algoritmizovaného postupu v podobě příkazů těchto zmíněných matematických programů má pro studenty, mezi nimi například budoucí programátory, či učitele matematiky nebo výpočetní techniky, poměrně velký význam. Uvědomí si, že počítačové programy se neobejdou bez algoritmů, za kterými velmi často stojí dobrá znalost matematiky a lidský důvtip, které dokázaly převést lidské metody řešení problémů do algoritmizované počítačové podoby.

Na druhou stranu ale také dojdou k poznání, že používání metody eliminace kvantifikátorů s sebou též nese jisté obtíže. Mezi ty můžeme zahrnout především výpočetní složitost. Je totiž nutné si uvědomit, že eliminace kvantifikátorů ve formuli o kupříkladu pěti proměnných má při výpočtech velkou paměťovou i časovou náročnost. Mimo jiné sem dále lze zařadit také například nutnost správného a přesného formulování příkazů zadávaných počítači. Jak jsme předvedli v úloze v kapitole 7.1.3, může i malá změna zápisu příkazu přinést diametrálně odlišný výsledek, jelikož takový příkaz bude počítačem interpretován jinak, než jej uživatel původně zamýšlel. I to je pro uživatele jistě nesporný přínos, ten je totiž nucen vyjadřovat se přesně a správně a je-li s fungováním programu i s možnými úskalími alespoň částečně obeznámen, má dobrou výchozí pozici, aby se mu to dařilo a počítač spolu s programy počítačové algebry se mu staly výkonnými pomocníky.

Jak již bylo řečeno, pole působnosti eliminace kvantifikátorů je hodně široké, jelikož do podoby kvantifikované formule obsahující soustavu rovnic a nerovnic lze převést nepřeberné množství matematických problémů, což jsme předvedli především v kapitole 7. Mezi jinými je to řešení rovnic a nerovnic a jejich soustav, vyšetřování průběhu funkcí, zjišťování existence kořenů polynomů a určování jejich hodnot, počítání průsečíků nejen rovinných křivek, dokazování matematických vět nebo dokonce řešení slovních úloh.

Na samotný závěr ještě dodejme, že samotný algoritmus cylindrické algebraické dekompozice, jehož několik různých verzí bylo v práci nastíněno, nemusí nutně sloužit jen k provedení eliminace kvantifikátorů. Nachází další uplatnění i v jiných oblastech matematiky, například při počítačových výpočtech dvojných a trojných integrálů lze CAD využít k určení částí roviny či prostoru, na nichž má být integrál počítán. Jako ukázku použijme následující příklad.

Příklad 8.1: Vypočtěte hodnotu dvojného integrálu $\iint_M 1\,dxdy$ na množině $M = \{\frac{x^2}{4} + \frac{y^2}{9} < 1\}$.

V podstatě se jedná o výpočet objemu eliptického válce o výšce 1, jehož podstava ležící v rovině dané osami x a y je určena vnitřkem elipsy o rovnici

$$\frac{x^2}{4} + \frac{y^2}{9} = 1.$$

„Lidský“ přístup k řešení by tedy spočíval v řešení tohoto dvojného integrálu pomocí jeho převedení na dvojnásobné integrály Fubiniovou větou. Pak by platilo

$$\iint_M 1\,dxdy = \int_{-2}^{2}\left(\int_{-3\sqrt{1-\frac{x^2}{4}}}^{3\sqrt{1-\frac{x^2}{4}}} 1\,dy\right)dx = \int_{-2}^{2} 6\cdot\sqrt{1-\frac{x^2}{4}}\,dx = 6\cdot\pi.$$

Výpočet můžeme nechat provést i program Mathematica, a sice pomocí příkazu **Integrate [Boole [x^2/4 + y^2/9 < 1], {y, –20, 20}, {x, –20, 20}]**, výsledek je 6π.

Celý postup v tomto případě spočívá v tom, že program pomocí příkazu **Boole [výraz]** nalezne oblast, na které je výraz, resp. rovnice či nerovnice, pravdivý, v našem případě se jedná o vnitřek zadané elipsy, a tedy i výstupní hodnota funkce **Boole** je pravdivá (jinými slovy této oblasti je přiřazena hodnota 1), tam, kde výraz pravdivý není, je výstup funkce **Boole** nepravdivý, hodnota přiřazená oblasti bude 0. Následuje již „pouhé“ integrování v zadané oblasti pro hodnoty

$$-20 \le x \le 20 \text{ a } -20 \le y \le 20.$$

Problém pro počítač nastává právě teď. Nemá totiž lidskou představu o elipse, která běžnému řešiteli dovolí napsat integrační meze téměř okamžitě. V programu Mathematica je, jak již víme, k dispozici algoritmus pro cylindrickou algebraickou dekompozici, takže ani počítač nemá s určením integračního oboru problém a meze

si nalezne. Zadáním příkazu **CylindricalDecomposition [x^2/4 + y^2/9 < 1, {x, y}]** získáme odpověď `−2 < x < 2 && −3/2 < y < 3/2`. To, že integrací dostaneme výše uvedený výsledek, nás již nepřekvapí. ♦

Jako poslední příklad dalšího užití uveďme jednu z úloh soutěže družstev šestého ročníku Středoevropské matematické olympiády pořádané od 6. do 12. září 2012 ve švýcarském Solothurnu.

Příklad 8.2: Nechť *a, b, c* jsou kladná reálná čísla, jejichž součin je roven 1. Dokažte, že platí nerovnost

$$\sqrt{9+16a^2}+\sqrt{9+16b^2}+\sqrt{9+16c^2}\geq 3+4(a+b+c).$$

V „lidském“ přístupu je nutné nejprve provést jistou přípravu, než se pustíme do důkazu samotného.

Dokážeme, že pro tři kladná reálná čísla *x, y, z*, jejichž součet je menší než 3, platí nerovnost

$$\frac{9-x^2}{x}\cdot\frac{9-y^2}{y}\cdot\frac{9-z^2}{z}>512.$$

Využijeme nerovnosti mezi aritmetickým a geometrických průměrem čtyř kladných reálných čísel a píšeme

$$3+x=1+1+1+x\geq 4\sqrt[4]{x},$$

stejně tak platí $3+y\geq 4\sqrt[4]{y}$ a $3+z\geq 4\sqrt[4]{z}$. Násobením levých a pravých stran těchto tří nerovnic dostáváme nerovnost

$$(3+x)(3+y)(3+z)\geq 64\sqrt[4]{xyz}.$$

Použijme ještě jednou nerovnost mezi aritmetickým a geometrickým průměrem, tentokráte pro tři proměnné, potom platí

$$3>x+y+z\geq 3\sqrt[3]{xyz}$$

a tedy

$$1>xyz$$

a

$$xy+yz+xz\geq 3(xyz)^{\frac{2}{3}}.$$

Násobíme-li mezi sebou výrazy $(3-x)(3-y)(3-z)$, dostáváme výsledek

$$9(3-x-y-z)+3(xy+yz+xz)-xyz>9(xyz)^{\frac{2}{3}}-xyz>8(xyz)^{\frac{2}{3}}.$$

Vynásobením levých a pravých stran nerovností $(3 + x)(3 + y)(3 + z) \geq 64\sqrt[4]{xyz}$ a $(3 - x)(3 - y)(3 - z) > 8\left(xyz\right)^{\frac{2}{3}}$ obdržíme nerovnost

$$(9 - x)(9 - y)(9 - z) > 512xyz,$$

tedy

$$\frac{9 - x^2}{x} \cdot \frac{9 - y^2}{y} \cdot \frac{9 - z^2}{z} > 512\,.$$

Nyní přikročme již k samotnému důkazu. Proměnné *x*, *y*, *z* definujme následujícím způsobem:

$$x = \sqrt{9 + 16a^2} - 4a\,,$$

$$y = \sqrt{9 + 16b^2} - 4b\,,$$

$$z = \sqrt{9 + 16c^2} - 4c\,,$$

po dosazení do levé strany nerovnice dostáváme

$$\frac{9 - x^2}{x} \cdot \frac{9 - y^2}{y} \cdot \frac{9 - z^2}{z} = 512abc = 512.$$

Tím je však porušena podmínka $\frac{9 - x^2}{x} \cdot \frac{9 - y^2}{y} \cdot \frac{9 - z^2}{z} > 512$, to je možné pouze tehdy, není-li pravdivý její předpoklad. Tím docházíme ke sporu, musí platit nerovnost

$$x + y + z \geq 3$$

a po dosazení za proměnné též

$$\sqrt{9 + 16a^2} + \sqrt{9 + 16b^2} + \sqrt{9 + 16c^2} \geq 3 + 4\left(a + b + c\right).$$

Tím je úloha vyřešena. [IMO12, SVR12]♦

Jak je vidět, „lidské“ řešení této úlohy není nikterak jednoduché a je nutné mít i nápad, jakým směrem postupovat, aby se v důkazu vůbec mohlo postoupit dál. Naproti tomu program Wolfram Mathematica je s důkazem hotový poměrně rychle, jestliže mu zadáme příkaz ve tvaru **Resolve [ForAll [{a, b, c}, a > 0 && b > 0 && c > 0 && a*b*c = 1, Sqrt [9 + a^2] + Sqrt [9 + b^2] + Sqrt [9 + c^2]]]**, na který počítač odpoví `True`. Jako poslední ještě udělejme to, že využijeme příkaz **Timing []** s argumentem odpovídajícímu výše uvedenému příkazu. Po jeho zadání zjistíme, že nalezení řešení trvalo programu poměrně krátkou dobu, řádově sekundy.

V žádném případě ale nejde říci, že výše popsané postupy řešení zadaných problémů by měly být upřednostňovány před metodami tradičními, i když lze předpokládat, že postupem času bude stále častěji docházet k využívání nejrůznějších matematických programů v procesu vzdělávání ve stejné míře, v jaké se v současné době využívají například kapesní kalkulátory, ale můžeme na ně nahlížet také jako na další možnost, jak se dobrat výsledku i v případě, že ostatní řešitelské postupy buď selhaly, nebo jsou příliš složité a časově náročné. Navíc je nejen v učitelské profesi vždy dobré mít v záloze způsob kontroly, že tradičními metodami dosažené výsledky úlohy jsou správné.

Mezi výše jmenovanými obecnými přínosy práce lze její hlavní přínos spatřovat také ve využitelnosti její jak teoretické, tak i praktické části v nezanedbatelném množství předmětů vyučovaných nejen v navazujícím magisterském studiu v oboru *Učitelství matematiky pro základní školy* na Katedře matematiky, fyziky a technické výchovy. Zde jsou studenti v rámci série předmětů Metody řešení matematických úloh postupně seznamováni mimo jiné s metodami důkazů matematických tvrzení, se způsoby řešení rovnic, nerovnic a jejich soustav, a to včetně rovnic, nerovnic a soustav s parametry, nebo v neposlední řadě s řešitelskými přístupy v problematice slovních úloh. Ve všech těchto oblastech lze eliminaci kvantifikátorů nejen dosti efektivně využít k řešení zadaných úloh, ale zároveň spolu s tím studentům předvést i novější trendy řešení spojené s využitím prostředků výpočetní techniky.

Literární a elektronické zdroje

[ACM84a] ARNON, D. S., COLLINS, G. E., MCCALLUM, S. Cylindrical algebraic decomposition I: the basic algorithm. *SIAM Journal on Computing*. 1984, vol. 13, is. 4, s. 865-877. Dostupný z: http://www.cs.purdue.edu/research/technical_reports/1982/TR%2082-427A.pdf

[ACM84b] ARNON, D. S., COLLINS, G. E., MCCALLUM, S. Cylindrical algebraic decomposition II: an adjacency algorithm for the plane. *SIAM Journal on Computing*. 1984, vol. 13, is. 4, s. 878-888. Dostupný z: http://www.cs.purdue.edu/research/technical_reports/1982/TR%2082-428.pdf

[BEE04] BEESON, Michael. The Mechanization of Mathematics. TEUSCHER. *Alan Turing: Life and Legacy of a Great Thinker*. Berlin: Springer Verlag, 2004. ISBN 3-540-20020-7. Dostupné z: http://csclub.cs.sjsu.edu/~beeson/Papers/turing2.pdf

[BRO02] BROWN, Christopher W. *QEPCAD – Quantifier Elimination by Cylindrical Algebraic Decomposition* [online]. United States Naval Academy, [2002] [cit. 2012-05-14]. Dostupné z: http://www.usna.edu/cs/~qepcad/B/QEPCAD.html

[BRO03]BROWN, Christopher W. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*. 2003, vol. 37, no. 4, s. 97-108.

[CHE11] CHEN, Changbo. *Solving Polynomial Systems via Triangular Decomposition*. London (Western Ontario, Canada), 2011. Dostupné z: http://www.csd.uwo.ca/~moreno/Publications/Changbo.Chen-Thesis-UWO.pdf. Disertační práce. The University of Western Ontario. Vedoucí práce Dr. Marc Moreno Maza.

[DAV09] DAVÍDEK, Ondřej. *Cylindrická algebraická dekompozice a její aplikace*. Plzeň, 2009. Diplomová práce. ZČU v Plzni. Vedoucí práce Ing. Bohumír Bastl, PhD.

[DUP82] DUPAČOVÁ, Jitka. *Lineární programování*. Praha: Státní pedagogické nakladatelství, 1982.

[FEF06] FEFERMAN, Solomon. Tarski's influence on computer science. *Logical Methods in Computer Science*. 2006, roč. 2, č. 3, s. 13. Dostupné z: http://arxiv.org/pdf/cs.GL/0608062.pdf

[HON08] HONZÍK, Lukáš. Možnosti využití eliminace kvantifikátorů v tělese reálných čísel. In *Žilinská didaktická konferencia 2008*. Žilina: Žilinská univerzita v Žiline, 2008. s. 1-5. CD-ROM. ISBN 978-80-8070-863-4.

[HOR09] HORA, Jaroslav. O počítačovém dokazování nerovností. In: *Žilinská didaktická konferencia 2009*. Žilina: Žilinská univerzita v Žiline, 2009. s. 1-6.

[HOP] HORA, Jaroslav, PECH, Pavel. O neobvyklé metodě výpočtu limit racionálních lomených funkcí v programu Mathematica.

[IMO12] Problems and Solutions MEMO 2012. In: *IMOSuisse* [online]. 2012 [cit. 2012-10-12]. Dostupné z: http://www.imosuisse.ch/skripte/MEMO/memo2012_solutions.pdf

[MAP12] MAPLESOFT. *Maplesoft – Online Help* [online]. Waterloo (Ontario, Kanada), 2012 [cit. 2012-05-15]. Dostupné z: http://www.maplesoft.com/support/help

[MAS04] MAŠEK, J. Přednášky z předmětu Počítačem podporovaná výuka. 2004.

[OCR03] O'CONNOR, J. J., ROBERTSON, E. F. *Tarski biography* [online]. c2003 [cit. 2009-07-16]. Dostupný z: http://www.gap-system.org/~history/Biographies/Tarski.html

[WTC09] Přispěvatelé Wikipedie. Alfred Tarski [online]. c2009 [cit. 2009-07-16]. Dostupný z: http://cs.wikipedia.org/w/index.php?title=Alfred_Tarski&oldid=3605665

[SVR12] ŠVRČEK, J. 6. Středoevropská matematická olympiáda. *Rozhledy matematicko-fyzikální: časopis pro studující středních škol a zájemce o matematiku, fyziku a informatiku*. Praha: Jednota českých matematiků a fyziků, 1921-, roč. 87, č. 4, s. 46-50. ISSN 0035-9343.

[TAR57] TARSKI, Alfred. *A Decision Method for Elementary Algebra and Geometry*. 2. vyd. Santa Monica, California: The RAND Corporation, 1957. Dostupné z: http://www.rand.org/pubs/reports/2008/R109.pdf

[WTE09] Wikipedia contributors. Alfred Tarski [online]. c2009 [cit. 2009-07-16]. Dostupný z: http://en.wikipedia.org/w/index.php?title=Alfred_Tarski&oldid=293569777

[WHI06] Wikipedia contributors. History of virtual learning environments. [online]. c2006 [cit. 2011-01-19]. Dostupný z: http://en.wikipedia.org/wiki/History_of_virtual_learning_environments.

[WIN96] WINKLER, Franz. *Polynomial Algorithms in Computer Algebra*. New York: Springer, 1996. 270 s. ISBN 32-118-2759-5.

[WOL12] WOLFRAM RESEARCH, Inc. *Wolfram Mathematica: Wolfram Mathematica 8 Documentation* [online]. Champaign (Illinois, USA), 2012 [cit. 2012-04-01]. Dostupné z: http://reference.wolfram.com/mathematica/guide/Mathematica.html

Seznam obrázků

Seznam publikací

HONZÍK, L. Možnosti využití eliminace kvantifikátorů v tělese reálných čísel. In *Žilinská didaktická konferencia 2008*. Žilina : Žilinská univerzita v Žiline, 2008. s. 1-5. CD-ROM. ISBN 978-80-8070-8.

HONZÍK, L. Počítačové metody eliminace kvantifikátorů v reálných uzavřených tělesech a možnosti jejich využití. In *Konference ICTE – Junior*. České Budějovice : Jihočeská univerzita v Českých Budějovicích, 2008. CD-ROM. ISBN 978-80-7394-107-9.

HONZÍK, L. Počítačové metody eliminace kvantifikátorů v *R* a možnosti jejich využití. In *Information and Communication Technology in Education, Ph.D. student's section*. Ostrava: University of Ostrava, 2009. s. 67-83. CD-ROM. ISBN 978-80-7368-460-0.

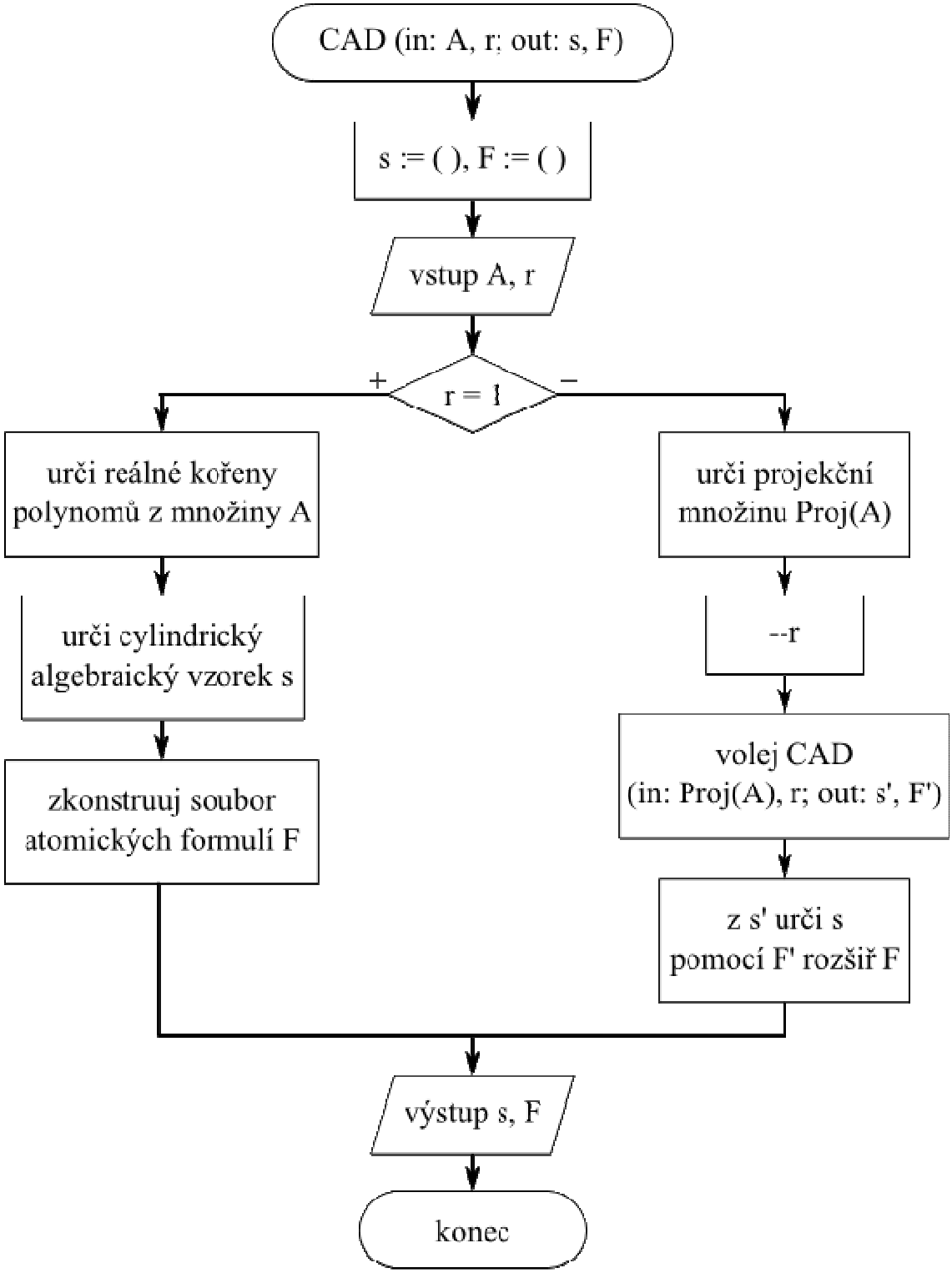
HONZÍK, L., TICHÝ, M. Geogebra - více než dynamická geometrie. *Matematika - fyzika - informatika*. 2010, roč. 19, č. 7, 8, s. 426-433, 499-507. ISSN 1210-1761.

HONZÍK, L. Několik úloh řešených Dirichletovým principem. In *Cieľom vyučovania matematiky je šťastný človek*. Žilina : Žilinská univerzita v Žiline, 2011. s. 213-219. ISBN 978-80-554-0393-9.

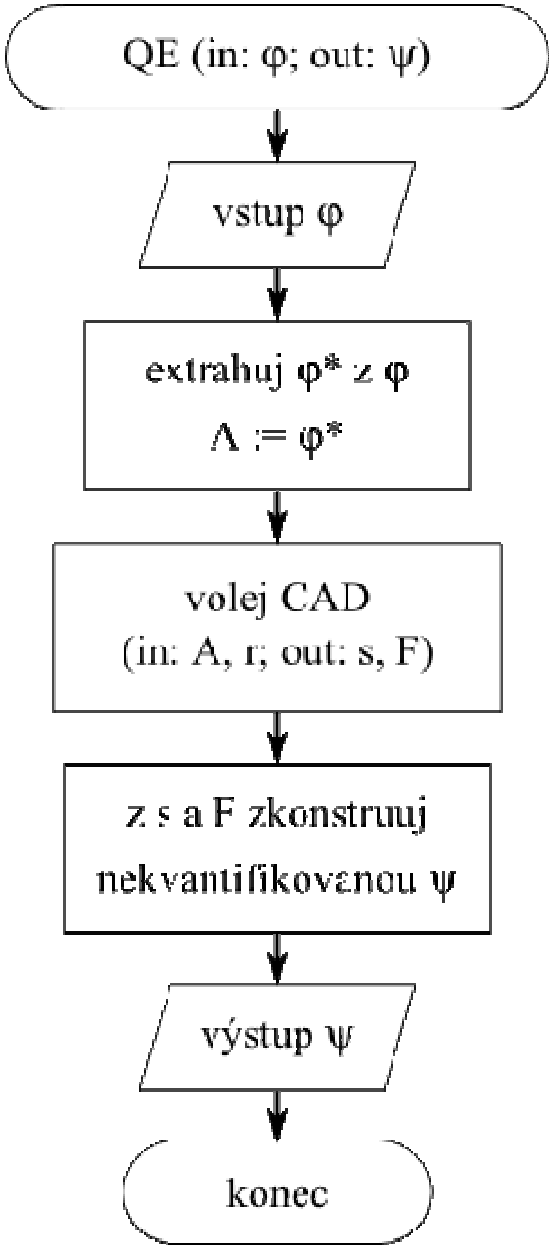
HONZÍK, L. Wolfram|Alpha. *Matematika - fyzika - informatika*. (odesláno, zatím kvůli organizačním změnám v redakci časopisu bez vyjádření redakce o uveřejnění)

HONZÍK, L. Využití eliminace kvantifikátorů v řešení jednoduchých optimalizačních úloh. *Journal of technology and information education*. (odesláno, zatím bez vyjádření redakční rady)

Příloha 1 – Vývojový diagram algoritmu cylindrické algebraické dekompozice



Příloha 2 – Vývojový diagram algoritmu eliminace kvantifikátorů s použitím CAD



Příloha 3 – Kompletní výpis příkazu *CylindricAlgebraicDecompose* ([$x^2 + y^2 - 1$], PolynomialRing ([x, y])) v programu Maple

$$F := [x^2 + y^2 - 1]$$

$$[x^2 + y^2 - 1]$$

$$R := PolynomialRing([x, y])$$

$$polynomial\_ring$$

$$CylindricalAlgebraicDecompose(F, R)$$

$$\left\{ \begin{array}{ll} [regular\_chain, [[-2, -2], [0, 0]]] & y < -1 \\ \left\{ \begin{array}{ll} [regular\_chain, [[-1, -1], [-1, -1]]] & x < 0 \\ [regular\_chain, [[-1, -1], [0, 0]]] & x = 0 \\ [regular\_chain, [[-1, -1], [1, 1]]] & 0 < x \end{array} \right. & y = -1 \\ \left\{ \begin{array}{ll} [regular\_chain, [[-2, -2], [-1, -1]]] & x < RootOf(\_Z^2 + y^2 - 1, index = real_1) \\ [regular\_chain, [[-1, -1], [-1, -1]]] & x = RootOf(\_Z^2 + y^2 - 1, index = real_1) \\ [regular\_chain, [[-1, -1], [1, 1]]] & And(RootOf(\_Z^2 + y^2 - 1, index = real_1) < x, x < RootOf(\_Z^2 + y^2 - 1, index = real_2)) \\ [regular\_chain, [[-1, -1], [2, 2]]] & x = RootOf(\_Z^2 + y^2 - 1, index = real_2) \\ [regular\_chain, [[-2, -2], [2, 2]]] & RootOf(\_Z^2 + y^2 - 1, index = real_2) < x \end{array} \right. & And(-1 < y, y < 1) \\ \left\{ \begin{array}{ll} [regular\_chain, [[0, 2], [-1, -1]]] & x < 0 \\ [regular\_chain, [[0, 2], [0, 0]]] & x = 0 \\ [regular\_chain, [[0, 2], [1, 1]]] & 0 < x \end{array} \right. & y = 1 \\ [regular\_chain, [[3, 3], [0, 0]]] & 1 < y \end{array} \right\}$$