VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Implementace jazyka TIL-Script

Implementation of the TIL-Script Language

Bc. Filip Peterek

Diplomová práce

Vedoucí práce: prof. RNDr. Marie Duží, CSc.

Ostrava, 2023

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/22/078945
vsboee2202603F

Zadání diplomové práce

Student: **Bc. Filip Peterek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Implementace jazyka TIL-Script
Implementation of the TIL-Script Language

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je implementovat funkcionální jazyk TIL-Script. Implementace bude vycházet z gramatiky jazyka, která je v souladu s logickým systémem Transparentní Intensionální Logiky (TIL).

Práce bude obsahovat:
1. Popis systému TIL, tj. jazyk konstrukcí a rozvětvená teorie typů.
2. Definici jazyka TIL-Script.
3. Analýzu, návrh a implementaci jazyka TIL-Script včetně typové kontroly.
4. Dokumentace celého projektu včetně uživatelské příručky.

Implementace bude provedena v jazyce C# nebo Java.

Seznam doporučené odborné literatury:

[1] Duží M., Materna P. (2012): TIL jako procedurální logika (průvodce zvídavého čtenáře Transparentní intensionální logikou). Aleph Bratislava 2012, ISBN 978-80-89491-08-7
[2] Duží M., Jespersen B. and Materna P. (2010): Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Transparent Intensional Logic. First edition. Berlin: Springer, series Logic, Epistemology, and the Unity of Science, vol. 17, ISBN 978-90-481-8811-6.
[3] Ciprich N., Duží, M., Košinár M. (2009): The TIL-Script Language. Frontiers in Artificial Intelligence and Applications, Amsterdam: IOS Press, vol. 190, pp. 166-179.

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

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

IČ: 61989100
DIČ: CZ61989100

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

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

Vedoucí diplomové práce: **prof. RNDr. Marie Duží, CSc.**

Datum zadání: 01.09.2022

Datum odevzdání: 30.04.2023

Garant studijního oboru: prof. RNDr. Václav Snášel, CSc.

V IS EDISON zadáno: 07.11.2022 11:59:22

Abstrakt

Cílem práce je implementovat programovací jazyk TILScript. Jazyk TILScript slouží jako výpočetní varianta logického kalkulu TIL, jenž umožňuje jednoduchý strojový zápis konstrukcí Transparentní intenzionální logiky, ale také jejich následné provedení. Práce dále řeší praktické problémy s interpretací TILScriptu, a to například definice pojmenovaných funkcí, interakce s databází, apod. Dále se práce snaží navrhnout nadmnožinu jazyka TILSkript, která umožní konstrukce TILu nejen provádět, ale také analyzovat, vytvářet je, a pracovat s nimi.

Klíčová slova

Transparentní intenzionální logika, TILScript, překladač

Abstract

The goal of the thesis is the definition and implementation of the TILScript language. TILScript is a scripting language which serves the purpose of a computational variant of Transparent intensional logic, a logical calculus based on typed lambda calculi. TILScript allows for not just representation, but also execution of TIL constructions. This work also deals with practical problems of TILScript implementation, such as definitions of named functions, interaction with databases, and so on. Furthermore, this thesis attempts to define a superset of the TILScript language, which allows for not just the execution of constructions, but also for their creation and analysis.

Keywords

Transparent intensional logic, TILScript, interpreter

Poděkování

TODO: Doplnit poděkování, až bude práce hotová

Obsah

Seznam použitých zkratek a symbolů

TIL	–	Transparentní intenzionální logika
JVM	–	Java Virtual Machine

Seznam obrázků

Seznam tabulek

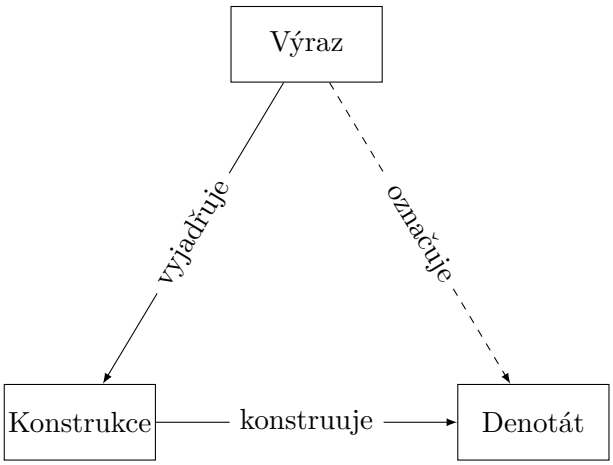
Kapitola 1

Transparentní intenzionální logika

Transparentní intenzionální logika (TIL) je logický systém založený na typovaném lambda kalkulu. TIL je využíván k logické analýze přirozeného jazyka. Oproti tradičnímu lambda kalkulu, jenž se využívá jako komputační model, tedy jako pouhý prostředek k dosažení konkrétní hodnoty – výsledku, v Transparentní intenzionální logice hraje konstrukce kalkulu často důležitější roli, než hodnota, kterou by konstrukce po provedení zkonstruovala.

Jako příklad využití lambda kalkulu jako výpočetní model lze uvést např. funkcionální programovací jazyk Haskell. Interně je Haskell kompilován do lambda kalkulu (přesněji do jeho supersetu obsahujícího např. čísla nebo logické hodnoty, která jinak v lambda kalkulu musíme kódovat pomocí Churchova kódování – K-kombinátorů, apod.). Ultimátně v Haskellu ovšem lambda kalkul slouží pouze k získání konkrétního výsledku. Nadefinujeme vztah mezi vstupem a výstupem, a program napsaný v Haskellu nám vstup transformuje. Pokud zanedbáme efektivitu programu, nezajímá nás, jakým způsobem program spočítal výsledek, dokud jej spočítal správně.

Naopak Transparentní intenzionální logika je hyperintenzionálním kalkulem, který nám umožňuje vytvářet konstrukce vypovídající o jiných konstrukcích. TIL vychází z myšlenky, že výraz přirozeného jazyka sice označuje denotát – konkrétní individuum, významem výrazu ovšem není samotný denotát, který ani nemusí nutně existovat. Význam výrazu je abstraktní a lze jej zachytit konstrukcí. Daná konstrukce poté při provedení zkonstruuje denotát výrazu. Jako příklad lze uvést například výraz "francouzský král."V době psaní této práce Francie krále nemá. Výraz nemá žádný denotát, neuvádí žádné konkrétní individuum. Přesto výrazu "francouzský král"rozumíme, výraz má svůj význam, jen v současné době neuvádí žádnou osobu. A budeme-li chtít o významu výrazu "francouzský král"něco vypovědět, například že francouzský král je monarchou v čele Francie, daný monarcha nemusí existovat. Dále lze uvést například rozdíl mezi výrazy "logaritmus 1024 o základě 2"a "5 + 5". Denotátem obou výrazů je 10. Zadáme-li do interpreteru Haskellu výrazy `logBase 2 1024` a `5 + 5`, získáme v obou případech stejný výsledek. V přirozeném jazyce ovšem chápeme značný rozdíl mezi oběma výrazy, ačkoliv mají stejný denotát. "Logaritmus 1024 o základě 2"vyja-



Obrázek 1.1: Schéma procedurální sémantiky TIL

dřuje číslo, kterým musíme umocnit dvojku, abychom získali 1024. Výraz "5 + 5"očividně vyjadřuje úplně jinou matematickou operaci a jeho výsledek spočítáme jiným postupem.

Denotátem výrazu může být nejen objekt z báze, ale i konstrukce nebo funkce.

Jak již bylo zmíněno, Transparentní intenzionální logika vychází z typovaného lambda kalkulu, proto také každý objekt musí mít svůj typ. Pro správné pochopení TILu, a tedy i této práce, je tak nutné znát typovou hierarchii TIL.

1.1 Báze

Báze je kolekce vzájemně disjunktních neprázdných množin, které dohromady definují s jakými objekty budeme pracovat. Tyto množiny definují atomické objekty. Každá množina dále objektům určuje určitá základní kritéria (např. pokud jako jednu z množin báze zvolíme množinu $\mathbb{N}$, víme, že všechny objekty z této množiny budou čísla). Vždy se ovšem jedná pouze o nejnutnější a nejzákladnější vlastnosti. Báze například nemá žádný vliv na vlastnosti proměnlivé v čase.

Bázi volíme dle potřeb konkrétní aplikace a univerza diskurzu. Například používáme-li TIL k logické analýze matematických vět, jako bázi lze zvolit například množinu celých čísel, množinu reálných čísel, a množinu pravdivostních hodnot. Musíme však vzít v potaz, že tato báze neobsahuje čísla komplexní.

Patří-li objekt x do množiny α z báze, říkáme, že se jedná o objekt typu α. K explicitnímu uvedení typu objektu x využíváme zápis x/α. Množinám tvořícím bázi lze přirozeně říkat typy.

Pro analýzu přirozeného jazyka se většinou volí objektová báze skládající se z typů o, ι, τ, ω. Tyto typy jsou podrobněji popsány v tabulce 1.1.

Tabulka 1.1: Výchozí báze pro analýzu přirozeného jazyka

Typ	Popis typu
o	Množina pravdivostních hodnot
ι	Množina individuí (univerzum diskurzu)
τ	Množina časových okamžiků/reálných čísel
ω	Množina logicky možných světů

1.2 Funkce

V matematice se jako základní molekulární typ využívají relace. Funkce je poté speciální typ relace, který je zprava jednoznačný. V TIL je však základním molekulárním typem funkce. Chceme-li v TIL vyjádřit n-ární relaci nad množinou $\alpha_1 \times ... \times \alpha_n$, lze tak samozřejmě udělat definicí n-ární funkce z $\alpha_1 \times ... \times \alpha_n$ do o, která každému prvku z $\alpha_1 \times ... \times \alpha_n$ přiřadí pravdivostní hodnotu na základě toho, zda prvek do relace patří, nebo ne.

Narozdíl od tradičního lambda kalkulu, kde jsou funkce pouze nulární nebo unární, v Transparentní intenzionální logice není arita funkce omezena. Dále můžeme v TIL pracovat s funkcemi parciálními.

1.2.1 Intenze a extenze

V TIL dále rozlišujeme funkce na tzv. *intenze* a *extenze*. Intenze jsou funkce z možných světů. Extenze jsou funkce, jejichž doménou množina možných světů není, a tudíž jejich funkční hodnota nezávisí na stavu světa.

Intenze jsou obecně funkce typu $(\alpha\omega)$ pro libovolný typ α. Nejčastěji se však jedná o funkce typu $((\alpha\tau)\omega)$, tedy funkce zobrazující možné světy do chronologií objektů typu α.

1.3 Konstrukce TIL

Konstrukce v Transparentní intenzionální logice jsou abstraktní procedury. Tyto procedury jsou strukturované – nejedná se o množiny, mají pevně danou strukturu, a na uspořádání případných podprocedur záleží. Tyto konstrukce lze podle definovaných pravidel provést. Provedením konstrukce, získáme výstup, případně nezískáme nic, viz pravidla provedení konstrukcí 1.4. Konstrukce, které nekonstruují žádný výstup, nazýváme *nevlastní* (anglicky *improper*). V TIL pracujeme s šesti druhy konstrukcí.

Jak již bylo zmíněno, konstrukce můžou v TIL operovat nejen nad objekty, které nejsou konstrukcemi, tedy nad objekty z báze, ale také nad jinými konstrukce. Konstrukce však může operovat pouze nad konstrukcemi nižšího řádu, než je konstrukce samotná, viz 1.6. Každou podkonstrukci, kterou musíme provést při provedení konstrukce, nazýváme *konstituentem*. V TIL existuje šest různých druhů konstrukcí. Dvě atomické – mají pouze jeden konstituent, a to sebe samotné, a čtyři

molekulární. Atomickými konstrukcemi jsou *trivializace* a *proměnné*. Mezi molekulární konstrukce poté řadíme *kompozice*, *uzávěry*, *provedení* a *dvojí provedení*.

Proměnné jsou konstrukce, které na základě valuace v v-konstruují objekty. Skutečnost, že proměnná x v-konstruuje hodnotu typu α značíme $x \rightarrow_v \alpha$.

Trivializace pro libovolný objekt X konstruuje samotný objekt X. Konstrukce je v Transparentní intenzionální logice potřebná, neboť výchozím módem pro konstrukce je provedení. Samotná konstrukce X by tak byla automaticky provedena, a místo konstrukce X bychom dostali pouze její denotát. Pokud bychom chtěli zkonstruovat konstrukci X, musíme ji trivializovat. Tím se provede pouze konstrukce trivializace. A protože trivializace nemá jiný konstituent, než sebe samotnou, konstrukce X se tak neprovede. V literatuře se trivializace X tradičně značí ${}^0X$. Obzvlášť se používá také zápis $'X$. Tento zápis je poté využit i v jazyce TILScript. Trivializaci lze považovat za ekvivalent funkce `QUOTE` z jazyka Lisp. Trivializace taktéž bývá využívána ke konstruování hodnot, které nelze provést (objekty z báze, funkce) a tudíž je nelze zmínit netrivializované.

Kompozice je využívána k aplikaci funkcí. Kompozice $[XY_1...Y_m]$ značí aplikaci funkce konstruované konstrukcí X na argumenty zkonstruované konstrukcemi $Y_1, ..., Y_m$. Pokud konstrukce X konstruuje funkci f, všechny podkonstrukce $Y_1, ..., Y_m$ konstruují hodnotu, a je-li funkce f na daných argumentech definovaná, kompozice v-konstruuje funkční hodnotu na těchto argumentech. V opačném případě je kompozice nevlastní.

Uzávěr $\lambda x_1...x_m Y$ je konstrukce v-konstruující funkci. $x_1, ...x_m$ musí být navzájem různé proměnné, Y musí být konstrukcí. Konstruce uzávěru je velmi podobná abstrakci v lambda kalkulu. Narozdíl od lambda kalkulu však v TILu může uzávěr konstruovat funkce s aritou vyšší než jedna. Uzávěr nemůže být nikdy nevlastní, může však konstruovat tzv. *degenerovanou funkci*, tedy funkci, která je nedefinovaná na celém definičním oboru.

Provedení ${}^1X$ je konstrukce v-konstruující objekt konstruovaný konstrukcí X. Pokud je konstrukce X v-nevlastní, je provedení ${}^1X$ také v-nevlastní. Jelikož je však provedení výchozím módem pro objekty, většinou se neuvádí. Provést lze pouze konstrukce. Objekty z báze (tedy čísla, individua, apod...) či funkce nelze provést, jejich provedení nekonstruuje nic. Proto je potřeba tyto objekty vždy trivializovat.

Dvojí provedení ${}^2X$ je poslední z výčtu konstrukcí. Je-li X konstrukcí v-konstruující konstrukci Y, a v-konstruuje-li konstrukce Y objekt Z, pak ${}^2X$ v-konstruuje Z. V opačném případě je dvojí provedení ${}^2X$ v-nevlastní.

Jiné konstrukce v Transparentní intenzionální logice neexistují.

1.4 Pravidla provedení konstrukcí

1.4.1 Princip kompozicionality

1.5 Typy 1. řádu

Nechť B je báze. Pak:

i) Každá množina z báze B je atomický typ řádu 1 nad B.

ii) Nechť $\alpha, \beta_1, ..., \beta_m (m > 0)$ jsou typy řádu 1 nad B. Pak soubor všech m-árních parciálních funkcí $(\alpha\beta_1...\beta_m)$, tedy zobrazení z $\beta_1 \times ... \times \beta_m$ do α, je molekulární typ řádu 1 nad B.

iii) Nic jiného není typem řádu 1 nad bází B.

1.6 Rozvětvěná hierarchie typů

1.7 Charakteristické rysy TIL

Příloha A

Dlouhý zdrojový kód

```
#include <vector>
#include <algorithm>


template<typename T>
void sort(std::vector<T>& to_sort) {
    std::sort(to_sort.begin(); to_sort.end());
}
```

Listing A.1: Dlouhý zdrojový kód v jazyce C++ načtený s externího souboru