Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Tvorba interaktívnych príbehov ako plánovanie

bakalárska práca

2015

Martin Zvara

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Tvorba interaktívnych príbehov ako plánovanie

bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatka
Školiace pracovisko:	Katedra Informatiky
Školiteľ:	RNDr. Jozef Šiška, PhD.

Bratislava, 2015

Martin Zvara

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Zvara
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Tvorba interaktívnych príbehov ako plánovanie
Interactive storytelling through planning

Cieľ: Navrhnúť a implementovať systém pre tvorbu príbehov (storytelling) používajúci plánovanie ako hlavný nástroj na tvorbu rozvetvených dejových línií, ktorého najdôležitejšou súčasťou je vyhodnocovanie kvality príbehu.

Anotácia: Storytelling je oblasť umelej inteligencie venovaná automatickému generovaniu príbehov / zápletiek. Venuje sa širokému rozsahu problémov od generovania jednoduchých abstraktných zápletiek, až po plnohodnotné generovanie príbehov v (väčšinou) anglickom jazyku. Nelineárne, rozvetvené dejové línie umožňujú používateľom vybrať si svoje vlastné pokračovanie. Aj keď sa do popredia dostávajú hlavne vďaka interaktívnym multimédiálnym technológiám, existovali už aj predtým vo forme takzvaných "gamebookov", v ktorých si čitateľ mohol vybrať medzi rôznymi pokračovaniami príbehu pomocou rôznych odkazov na číslované odstavce alebo strany.

Kľúčové slová: Plánovanie, Tvorba príbehov, Umelá inteligencia

Vedúci: RNDr. Jozef Šiška, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, PhD.

Dátum zadania: 23.10.2014

Dátum schválenia: 28.10.2014 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Rád by som poďakoval:

Môjmu školiteľovi, RNDr. Jozefovi Šiškovi, PhD.,
za dobrú tému práce a za jeho nápady pre návrhu systému;

Rodičom,
za pomoc pri písaní práce.

Abstrakt

Táto bakalárska práca sa snaží uviesť čitaťela do problematiky výpočtovej naratológie so zameraním na systémy generujúce príbehy. Pojednáva o možnostiach ktoré sa poskytujú pri ich tvorbe a zároveň ponúka stručný prehľad niektorých významných systémov pre generovanie príbehov. Ukazuje ako sformalizovať Proppov model pre tvorbu príbehov a zapísať ho ako plánovací problém ktorý možno naprogramovať ako logický program pomocou ASP. Súčasťou je aj program ktorý generuje príbehy.

Kľúčové slová: systémy generujúce príbehy, plánovanie, ASP, Propp

Abstract

This bachelor thesis seeks to introduce reader to the topic of computational narratology, focusing on the systems generating stories. It discusses the opportunities that are provided for their creation and also offers a brief overview of some important systems generating stories. It shows how to formalize Propp model for creating stories and write it as a planning problem that can be programmed as a logic program using ASP. Also included is a program that generates stories.

Keywords: systems generating stories, planning, ASP, Propp

Obsah

Úvod

Vedci pôsobiaci v oblasti informačných technológií si neustále kladú rôzne otázky. Jedna z týchto otázok znie nasledovne: Je možné simulovať ľudskú kreativitu? Touto otázkou sa zaoberá vedná disciplína, počítačová kreativita, ktorá zahŕňa okruhy umelej inteligencie a kognitívnych vied. Jedným z mnoha odvetvý tejto vedy je výpočtová naratológia, ktorá sa zaoberá simuláciou kreatívnych procesov v oblasti literárnej tvorby.

Literárna tvorba obsahuje mnoho aspektov ktoré sa snaží výpočtová naratológia pochopiť a formalizovať. Pri tejto činnosti jej pomáha plánovanie. V kapitole "Storytelling a plánovanie" sa najprv oboznámime s plánovaní a spomenieme definície, ktoré neskôr budeme využívať. Potom spomenieme počiatky výpočtovej naratológie a ukážeme si prehľad významnejších systémov, ktoré generujú príbehy. Na koniec tejto kapitoly sa pozreme na systémy, ktoré genrujú netradičné, ale podstatné literárne aspekty.

Cieľom tejto bakalárskej práce je aj interakcia, ktorá je vbudovaná do systámov generujúcich príbehy. V kapitole "Interaktivita v príbehoch" poukážeme na hlavné typy interakcie a spomenieme ich výhody a nevýhody. Okrem toho si spomenieme na aké vlastnosti literárneho diela interakcia vplýva a ako by mala byť vbudovaná do systémov, aby bola v súhre z týmito vlastnosťami a tak neničila celkový dojem z diela.

V ďalšej kapitole z názvom "Prístupy k tvorbe príbehu" si opíšeme niektoré literárne postupy pre tvorbu príbehov a ukážeme si ich výhody a nevýhody použitia v rôznych programátorských prístupoch ku tvorbe literárneho diela.

Nasledujúca kapitola "Implementácia" sa zameria na formalizáciu Proppovej analýzy pomocou ASP a návrh systému, ktorý ju bude využívať pri tvorbe príbehov.

V poslednej kapitole "Výsledky" zdokumentujeme a rozoberieme príbehy ktoré vytvoril implementovaný systém.

Kapitola 1

Výpočtová naratológia a plánovanie

1.1 Plánovanie a ASP

Plánovanie je proces pri ktorom hľadáme akcie ktoré musia byť vykonané, aby sa dosiahlo požadovaného cieľa. Keďže sa plánovanie stalo jedným z hlavných nástrojov pri výpočtovej naratológii uvedieme si formálne definície pojmov ktoré neskôr budeme používať.

Definícia 1. *Fluent literal je atomická formula alebo jej negácia.*

Fluenty opisujú vlastnosti ktoré platia alebo neplatia v danom stave a možno ich v prvorádovej logike reprezentovať pomocou predikátov.

Definícia 2. *Plánovacia doména je trojica*

$$D = (S, A, \Phi)$$

kde S je množina stavov, A je množina akcií a Φ *: S x A* $\rightarrow$ *S je prechodová funkcia.*

Akcia sa skladá z efektov a predpokladov. Funkcia $\Phi(s,a) = s'$ hovorí, že vykonanie akcie a v stave s vedie k novému stavu s' ktorý obsahuje efekty akcie a.

Definícia 3. *Plánovací problém je trojica*

$$P = (D, Init, Goal)$$

kde D je plánovacia doména, Init $\subseteq$ *S je množina počiatočných stavov a Goal* $\subseteq$ *S je množina cieľových stavov.*

Definícia 4. *Plán v doméne D je postupnosť akcií* $a_1, a_2, \ldots, a_n$*;* $a_i \in A$*. Plán* $a_1, a_2, \ldots, a_n$ *je riešením plánovacieho problému P = (D,Init,Goal) práve vtedy, keď pre všetky* $s_0 \in$ *Init plaí*

$$\Phi(\ldots\Phi(\Phi(s_0, a_1), a_2)\ldots), a_n) \in Goal$$

Formálnejšie môžeme plánovanie popísať ako proces hľadania deduktívneho dôkazu tvrdenia. Tento proces je výpočtovo náročný. Problém hľadania plánu však možno redukovať na problém hľadania vyhovujúcej interpretácie pre množinu formúl výrokovej logiky. Plánovací problém vieme zapísať ako logický program. Túto formu programovania nazývame answer set programming alebo v skratke ASP. Logické pravidlá v ASP zapisujeme nasledovne:

Pravidlo:	A:-$B_0, B_1, \ldots, B_n$, not C_0, not $C_1, \ldots,$ not C_m.	$n, m \in \mathbb{N}^+$
Zákaz:	:-$B_0, B_1, \ldots, B_n$, not C_0, not $C_1, \ldots,$ not C_m.	A, B_i, C_j sú literály
Fakt:	A.	tvaru l,¬l

Výraz A nazývame hlava pravidla a výraz $B_0, B_1, \ldots, B_n$, not C_0, not $C_1, \ldots,$ not C_m. je telo. Literál not C_i nazývame default literál a je pravdivý v interpretácii I vtedy a len vtedy, keď $C_i \notin$ I. Intuitívne, hlava pravidla je pravdivá práve vtedy, keď je pravdiví každý literál v tele. Fakt je pravdivý vždy a zákaz slúži na zúženie pročtu množín ktoré sú riešením programu. Logický program zostrojený z takýchto pravidiel nazývame normálny logický program. Ak pravidlá v tele obsahujú len atómy, potom hovoríme o pozitívnom logickom programe. Riešením logického programu je stabilný model.

Definícia 5. *Interpretácia I je modelom logického programu P vtedy a len vtedy, keď každé pravidlo r* $\in$ *P je pravdivé v I.*

Definícia 6. *Nech I je interpretácia. Zúženie normálneho logického programu P je pozitívny logický program P' ktorý získame z P odstránením*

- *pravidiel obsahujúcich defaultový literál L taký, že platí* $I \not\models L$
- *defaultového literálu L takého, že platí* $I \models L$ *z ostávajúcich pravidiel*

Definícia 7. *Interpretácia I je stabilným modelom normálneho logického programu P vtedy a len vtedy, keď I je najmenším modelom zúženia P.*

Pri programovaní logického programu v ASP pre plánovací problém musíme vedieť reprezentovať fluenty, akcie, počiatočný a cieľový stav... teda objekty, ktorými ho popisujeme. Ak navyše program pracuje nad dynamickou doménou, nevyhneme sa niekoľkým problémom. Prvý problém súvisí s hľadaním postupnosti akcií, ktoré riešia plánovací problém. Pri hľadaní plánu musíme zabezpečiť generovanie akcií. Ďalej musíme riešiť otázku platnosti fluentov. Nech F_1 je fluent ktorý patrí stavu S_1. Po vykonaní akcie A_1 sa dostaneme do stavu S_2 ktorý bude obsahovať efekty akcie A_1. Ak efekty akcie A_1 nemenia platnosť fluentu F_1, má v stave S_2 platiť F_1? Problém, čo sa nemá zmeniť medzi stavmi S_i a S_{i+1} nazývame frame problem.

1.2 Výpočtová naratológia

Pojem naratológia bol zavedený bulharským historikom, filozofom a literárnym kritikom Cvetanom Todorovom, ktorý s nim označoval teóriu zameranú na narácie[1], rozdiely medzi nimi a zovšeobecnenie ich spoločných čŕt. Táto vedná disciplína zasahuje do oblastí umelej inteligencie zameriavajúcich sa na tvorbu inteligentných počítačových rozhraní a systémov simulujúcich ľudské správanie. Kôli potrebe princípov naratológie v oblasti informačných technológií sa začalo s ich algoritmizáciou a tak sa začala rozvíjať výpočtová naratológia. Pod pojmom výpočtová naratológia rozumieme teóriu, ktorá sa zaoberá skúmaním narácií z hľadiska ich spracovania pomocou výpočtovej

[1] príbeh alebo postupnosť udalostí

techniky. Zameriava sa na algoritmy ktoré vytvárajú a interpretujú narácie. Ďalší vplyv na výpočtovú naratológiu mala lingvistika. Jej hlavným prínosom v posledných rokoch sú najmä pokoročilé techniky "text mining"[2].

Výskum orientovaný na hľadanie generických algoritmov pre tvorbu narácií zameral svoju pozornosť na plánovanie. Systémy používajúce plánovanie pre tvorbu príbehu využívajú zväčša ruský formalizmus konštrukcie deja a to fabulu a sujet. Fabula je príbeh, ktorého udalosti sú podané v poradí akom sa odohrali a vedie práve jednu hlavnú dejovú líniu. Oproti tomu sujet je umelecky realizovaná fabula.

Pri tvorbe príbehu sujetovskej konštrukcie pomocou plánovania založeného na Aristotelovej definícií mýtu, ktorá delí dej na tri hlavné časti a udáva postavám ich ciele, môžeme udalosti zoradiť do istej postupnosti ktorá rozvinie príbeh z počiatočného stavu ku stavu konečnému. Tieto udalosti môžu a nemusia obsahovať skryté významy, ktoré nie sú spomenuté v príbehu. Nevýhodou tohoto prístupu je skutočnoť, že ľudia používajú značné množstvo informácií pri interpretácii príbehov. Môžeme to vidieť na krátkom príklade vety od anglického spisovateľa E. M. Forstera : „Kráľ umrel a kráľovna umrela od žiaľu.“, kde je každému jasné prečo bola kráľovna smutná avšak implementácia takéhoto ľudského chápania a tvorenia je veľmi zložitá. Ďalší problém nastáva pri tvorbe niektorých aspektov jazyka ktoré je ťažko formalizovať ako irónia, humor a mnoho ďalších. Takéto problémy nenastávajú pri koštrukcii fabuly.

1.3 Vývoj výpočtovej naratológie

Od sedemdesiatych rokov dvadsiateho storočia prešli systémy výpočtovej naratológie určitým rozvojom. V tejto podkapitole si spomenieme niektoré z nich.

[2]proces získavania dôležitých informacií z textu

1.3.1 Storytellingové systémy

V roku 1973 S. Klein so svojím tímom podal správu s názvom Automatic Novel Writing[8]. Tento systém sa opieral o mikrosimuláciu modelu, kde správanie jednotlivých chrakterov a udalosti, ktoré sa odohrali, boli založené na pravdepodobnostných pravidlách, ktoré menili stav simulovaného sveta. Kedže tu nebol určený jednoznačný cieľ, bolo potrebné zadefinovať pravidlá, ktorými sa generovanie riadilo.

Rozdielny prístup ku generovaniu mal interaktívny systém Talespin[15] ktorý simuloval racionálne spravanie postáv vo svete. Interaktivita spočínala v možnosti nastavenia počiatočného stavu sveta alebo výberu hesla zo zoznamu v duchu ktorého sa príbeh bude niesť. Tento raz už bol charakterom zadaný cieľ a pri jeho plnení mohlo vznikať viac podcieľov ktoré sa museli splniť. Talespin sa skladá z troch aktívnych častí, a to z jednotky na riešenie problémov, ukladacieho mechanizmu, ktorý ukladal jednotlivé udalosti do pamäte a tvoril celkovú databázu znalostí o svete, a vyvodzovacieho mechanizmus ktorí vyvodzoval dôsledky udalostí.

Ďalším v rade je systém Universe[9] ktorý sa nezameriaval na detailné opisy akcií, ale skôr na podanie konzistentného a zmysluplného príbehu s dostatkom informácií. Universe bol založený na plánovaní pričom okrem definovania cieľov postáv sa udali aj ciele autora ktoré mali väčšiu prioritu. Pri generovaní si systém udržiaval v pamäti dva precedenčné grafy, ktoré určovali vzťahy medzi rôznymi nesplnenými cieľmi autora a postáv a ich vzťah k odohratým udalostiam. Následne sa z týchto grafov vybral jeden autorov cieľ, ktorý sa rozvinul a tento postup sa rekurzívne opakoval pokiaľ sa nesplnil určitý počet cieľov.

Mexica[17] bol systém, ktorý generoval krátke príbehy o oddruženom kmeni Aztékov. Idea tvorby príbehu spočívala v inšpirácií z ostatných príbehov a následné vytvorenie príbehu podľa zistenej schémy. Systém na vstupe dostal príbehy v podobe textu, ktoré zanalyzoval a vytvoril na ich základe schému, ktorá sa následne použila ako vzor podľa ktorého sa má príbeh riadiť.

Prevratný bol aj systém Brutus[2] kde sa autori pozerali na tvorbu príbehu

ako na proces dokazovania teorém. Generovanie príbehu nebolo založené na simulácii sveta, ale na princípoch štruktúr, ktoré sa získali matematizáciou literárnych tém. Pri vytváraní deja sa určili postavy, ktoré mali pevne zadané ciele a mohli vykonávať akcie za istých podmienok.

Fabulist[19] je jedna z novších architektúr na generovanie príbehov. Používa plánovací algoritmus IPOCL[3] ktorý naraz rozhoduje o obvyklom chovaní postáv s určitým charakterom a ich motivácii. Výsledkom je zmysluplný príbeh s ľudskejšími postavamy.

1.3.2 Generovanie literárnej poézie a jej prvkov

Okrem tvorenia príbehov nám jazyk umožňuje tvoriť veľké množstvo literárnych foriem ako sú vtipy, metafory, básne a ďalšie. Keďže sú tieto formy navzájom odlišné potrebujú osobitný prístup a preto toto odvetvie nie tak dobre rozvinuté ako generovanie príbehov. Napriek tomu, aj tu sa dosiahlo úspechov ktoré stoja za povšimnutie.

Systém MIDAS[14] je schopný reprezentovať znalosti konvenčných metafor, interpretovať metafory pomocou týchto znalostí a učiť sa nové metafory. Okrem MIDAS existuje viac teórií a systémov ktoré sa zaoberajú interpretáciou metafor ale nie ich generovaním. O tvorbe metafor pojednáva Mark A. James[7] ktorý sa však zameriava na ich špecifickú triedu. Zaujímavejším je systém NETMET[22] ktorý vie interpretovať a tvoriť metafory. Samotný proces generovania spočíva vo vytvorení analogickej mapy zo zdroja zvolenej sémantickej oblasti ku vybratej téme a následnom vytvorení metafory.

Výpočtová naratológia može byť aj zábavná. Svedčia o tom systémy generujúce vtipy. Jedným z nich je JAPE[20] a je skôr určený pre deti. JAPE generuje vtipy na základe určitých pravidel, ktoré opisujú štruktúru humoru. HAHAcronym[23] je systém generujúci ironické akronymy[4]. Tento systém vie zanalyzovať akronymy a rozoznať ktoré slová v nich nechať nezmenené a ktoré je potrebné zmeniť.

[3] Intent-Driven Partial Order Causal Link

[4] slovo, názov zložený zo začiatočných písmen alebo slabík viacerých slov

Generovanie básní je náročnejší proces, pretože sa v nich využíva mnoho umeleckých vyjadrovacých prostriedkov, ktoré samé nie sú celkovo preskúmané. Aplikácia ASPERA[6] sa zameriava na tvorbu španielskych básní. Kombinuje NLG[5] a CBR[6] techniky ktoré aplikuje na konštrukcie získané heuristikami zo španielskej poézie.

[5]natural language generation
[6]case-based reasoning

Kapitola 2

Interaktivita v príbehoch

Interaktivivita je siliný nástroj pre zachitenie používateľovej pozornosti. Nie len že ho vtiahne do diania ale zároveň ho núti premýšľať. V tejto kapitole si rozoberieme možnosti interaktivity v systémoch generujúcich príbehy.

2.1 Hlavé postupy pri tvorbe interaktívnych príbehov

V druhej polovici dvadsiateho storočia sa začala dostávať interaktivita do tvorby príbehov v elektronickej podobe. Používala sa najmä v počítačových hrách a hypertextovej literatúre. Pri tvorbe týchto systémov sa autori museli vysporiadať z otázkou: Do akej hĺbky možno sprístupniť interaktivitu používateľovi? Každý tvorca sa vysporiadal s touto otázkou inak a preto vznikli rôzne spôsoby interakcie ktoré môžme zhrnúť pod štyri hlavné.

Diskurzívna interakcia

V podkapitole 1.2 sme si povedali, že príbeh možno podať dvoma spôsobmi a to ako fabulu a sujet. Tento typ interakcie neovplyvňuje príbeh samotný, ale skôr spôsob jeho podania používateľovi, čiže operuje s vlastnosťami sujetu.

Lokálna interakcia

Ako názov vypovedá, lokálna interakcia ovplyvňuje určitú časť príbehu. Po tomto type interakcie zväčša nasleduje preddefinovaná udalosť. Príkladom môže byť súboj alebo vyriešenie nejakého logického problému pre prostup v hre. V určitých prípadoch je potrebné aby uživaťel mal zručnosť pri riešení daných problémov a preto je lokálna interakcia zábavnejšia.

Vetvenie

Pri vetvení je príbeh reprezentovaný ako graf udalostí kde každý vrchol odpovedá používateľovej voľbe. Ak je graf orientovaný môže autor kontrolovať cestu ktorú bude príbeh nasledovať. Problém tohto prísupu je, že počet vetiev grafu narastá[1] exponenciálne a tak narastá množstvo možnotí, ktoré musí autor implementovať. Jedným riešením môže byť spájanie alebo odsekávanie vetiev. Používateľova voľba však stále neovplyvňuje príbeh samotný. Vylepšenie tohto postupu spočíva v ukladaní používateľových volieb a následnej úprave grafu čo v konečnom dôsledku pridáva pocit ovplyvňovania príbehu.

Neštrukturovaná interakcia

Neštrukturovaná interakcia je podobná vetveniu s rozdielom že graf možností nie je orientovaný. Je veľmi otázne či možno rozpávať o príbehu pretože používateľ môže prechádzať rôznymi udalosťami príbehu opakovane a tak môže stratiť smer, ktorým sa má príbeh odohrávať.

Z pohľadu rozsahu interaktivity si môžeme všimnúť, že najviac interaktívne sú počítačové hry žánru RPG alebo stratégie, avšak interakcia týchto hier neovplyvňuje príbeh samotný, zatiaľ čo menej interaktívne hry sú žánru adventúra. Možno tieto hry vskutočnosti považovať za interaktívne? Všimnime si zásadný rozdiel medzi rozprávaním príbehu a príbehom samotným. Rozprávanie príbehu je proces zatiaľ čo príbeh je súbor dát. Nie je možné interagovať z dátami ale je možné zasahovať do procesu. Preto by mala byť

[1] pri stromovej reprezentácii

interaktivita spoluprácou používateľa a systému, ktorý určitím spôsobom obmedzí používateľa[2] ale zároveň mu poskytne príjemný pocit pri interakcii z tvorby príbehu.

2.2 Základné vlastnosti interakcie v príbehoch

Príbeh vytvorený pomocou interaktívnych systémov tak isto ako literárne dielo vytvorené človekom musí spĺňať niektoré základné vlastnosti. Pri správnom využití týchto aspektov sa stáva konečný produkt ešte zaujímavejším. V tejto podkapitole si uvedieme päť z nich a povieme ako ich použiť.

2.2.1 Autorský zámer

Tento literárny aspekt odkazuje na myšlienku ktorú chcel autor v diele podať. Autorský zámer plne závisí len na autorovy a príbehu, ktorý chce zostrojiť. Všimnime si, že podanie autorského zámeru nemusí znamenať jeho doslovné, priame vloženie do diela. Naopak, môže byť vyjadrený cez udalosti ktoré sa v diele odohrali. Vzhľadom na danú definíciu by mali interakívne systémy podávať zmysel diela používateľovy a zároveň poskytovať autorovy dostatočnú kontrolu nad prvkamy, ktoré sú klúčové pri tvorbe diela.

2.2.2 Textová imerzia

Dôležitým prvkom v literárnom diele je jeho kvalita z pohľadu čitateľa. Túto vlastnosť nazývame textová imerzia, ktorú chápeme ako schopnosť používateľa vnoriť sa do deja. V interaktívnych systémoch sa imerzia vzťahuje na vlastnosť ako dobre dokáže interaktivita zabaviť používateľa podaním príbehu. Tieto systémy musia zvládnuť interaktivitu bez toho, aby vyrušili čiateľa alebo prerušili odohrávanie deja, pretože v opačných prípadoch takýto spôsob interakcie môže pôsobiť na úkor narácie.

[2]nechceme aby používateľ vytvoril príbeh sám

2.2.3 Vnútorná kompozícia diela

Vnútorná kompozícia diela je vo všeobecnosti postup akým je dielo podané príjemcovy. Pracuje s rytmom a formou literárneho diela. Interaktivita zvyčajne uvoľňuje dielo z fixnej kompozície, to však ale neznamená že ju dielo nemá mať. Uvedomme si dôležitosť vnútornej kompozície. Dobre spracovaný príbeh udržiava potrebu čitateľa v pokračovaní čítania. Preto hlavnou úlohou ineraktívneho systému pri generovaní príbehu je schopnosť riadiť vnútornú kompozíciu kôli poskytnutiu zážitku, ktorý vychádza z jej prvkov. Pri takejto funkcionalite sa systém nesmie obmedzovať na daný typ kompozície. Naopak, musí byť schopný vytvoriť štruktúru, ktorá zodpovedá generovanému príbehu a uchytí čitateľovu pozornosť.

2.2.4 Naratívna výpoveď

Naratívna výpoveď je úzko spätá s autorským zámerom. Sú to zachytené myšlienky a odpovede na daný problém v diele. Naratívna výpoveď kladie základnú otázku na interaktívne príbehy a to ako príbeh odpovedá používateľovi alebo naopak, aký má používateľ vplyv na príbeh. V interaktívnych systémoch, ktoré generujú príbehy môžno tieto otázky riešiť rôzne, pretože to závisí na forme interakcie ktorú systém poskytuje. Niektoré systémy poskytujú používateľom zvoliť si, čo sa stane v príbehu a ako skončí. Toto riešenie však pôsobí proti autorskému zámeru a textovej imrezii. Aby sme odstránili protichodnosť vlastností, môžme sprístupniť interakciu, ktorá ovplyvní príbeh samotný[3] a zároveň nechá stále volnosť systému pri tvorení príbehu. Preto by mali systémy umožnovať rôzne možnosti nastavenia ktoré budú mať v konečnom dôsledku vplyv na príbeh.

[3]napr. voľba prostredia v ktorom sa príbeh bude odohrávať

2.2.5 Riadenie deja

Riadenie deja je spojením všetkých predchádzajúcich vlastností a je úzko späté s interakciou. Je to ťažná sila príbehu ktorá by mala nechať používateľovy širokú škálu možností interakcie v každom okamihu rozvíjania príbehu a zároveň zohľadniť tento čiastočný vplyv. K významu čiastočnej interaktivity by sa malo pristupovať ako k jej schopnosti pomáhať vlastnostiam príbehu. Napríklad kôli imerzii by mal niektoré možnosti alebo nastavenia určiť systém samotný bez zásahu používateľa. Takto sa zároveň zabráni prerušeniu deja kôli nejakým rozhodnutiam, ktoré je potrebné spraviť pred jeho pokračovaním. Ideálny interaktívny systém pre generovanie príbehov by mal byť schopný poskytovať veľkú škálu možností ktoré môže používateľ vedome využiť a následne nechať systém vytvoriť príbeh.

Ako sme si ukázali interakcia môže byť silnou zbraňou, ktorú môžu využívať nie len systémy generujúce príbehy, avšak treba s ňou narábať opatrne, pretože môže byť tak isto škodlivá ako užitočná.

Kapitola 3

Prístupy k tvorbe príbehu

Keďže schopnosť rozoznania kvality príbehu je subjektívna vec, neexistuje jednotný prístup k jeho tvorbe. V tejto kapitole si ukážeme možnosti, ktoré sa nám poskytujú na tvorbu dobrého príbehu.

3.1 Literárne prístupy

Veľká odlišnosť literárnych diel si vynútila ich kategorizáciu. Každé dielo vieme zaradiť pod poéziu alebo prózu. Prózu vieme ďalej triediť podľa literárnych žánrov a štýlov. Každý žáner a štýl má svoje špecifické črty podľa ktorých vieme určiť jednotlivé literárne diela. Keď si odmyslíme toto delenie a pozrieme sa na prozaické dielo ako také, zistíme že je to podanie rôznych udalostí, ktoré sú nejak prepojené v istom slede. Aby dielo bolo úspešné, musia udalosti ktoré sa v ňom vyskytujú pôsobyť na osobu ktorej sú podané a to tak, aby v nej vyvolávali nejaké pocity. Ľudia chceli nájsť jednotný prístup ktorý spĺňa tieto podmienky a preto už od starovekého Grécka skúmali naratívne diela a dospeli k pravidlám, ktoré by mali tieto štruktúry spĺňať.

3.1.1 Freytagova analýza

Gustav Freytag bol nemecký spisovateľ, ktorý sa zaoberal najmä dramatickou tvorbou. V roku 1863 napísal knihu s názvom Technika drámy, kde uviedol svoju analýzu dramatických diel neskôr známu ako Freytagova pyramída [10]. Podľa Freytaga by sa malo dramatické dielo skladať z týchto častí:

1. **Expozícia** je úvodom do deja. Vysvetľujú sa v nej dôležité informácie a uvádzajú sa postavy.
2. V **kolízii** nastane udalosť ktorá ovplyvní celý dej.
3. **Kríza** je vyvrcholenie všetkých udalostí ktoré nastali od kolízie.
4. **Peripetia** alebo rozuzlenie je vyriešenie hlavného konfliktu.
5. V **závere** sa objasnia niektoré nedoriešené otázky.

Medzi kolíziou a krízou by malo napätie stúpať a naopak medzi krízou a peripetiou by malo napätie klesať. Z toho môžeme usúdiť, že jednotlivé udalosti by mali byť dejové zvraty. Takéto rozdelenie hovorí najmä o vplyve udalostí na osobu, ale neurčuje presne tieto udalosti.

3.1.2 Trojaktová štruktúra

Autorom tohoto modelu je americký scenárista Syd Field, ktorý rozdelil písanie scenáru do troch aktov a to na **prípravu**, **spor** a **vyjasnenie**. Tieto akty nehovoria presne o akciách, ktoré v nich majú byť vykonané, ale určujú čo by sa v nich malo dosiahnuť. Tiež je v nich určené stúpanie a klesanie napätia. Táto štruktúra priamo nevychádza zo skúmania literárnych diel, aj keď niektoré jej prvky možno nájsť vo svetovej literatúre. Oproti Freytagovi je tento prístp striktnejší a preto je niektorými scenáristami odsudzovaný.

3.1.3 Proppove funkcie

Vladimir Jakovlevič Propp bol ruský folklórista, ktorý sa zaoberal skúmaním naratívnych textových štruktúr, ich porovnávaním a pôvodom rozprávok. V roku 1928 napísal knihu s názvom Morfológia rozprávok [18], v ktorej okrem roztriedenia žánrov rozprávok definoval tridsaťjedna funkcií, ktoré sa dajú zhrnúť celkovo do šiestich funkcií a sedem hlavných charakteristík postáv, ktoré obsahuje väčšina literárnych diel. Tento prístup je zo všetkých spomenutých najstriktnejší, pretože Proppove funkcie presnejšie určujú aké udalosti sa majú v deji nachádzať a tak nenechávajú voľnosť autorovi.

Medzi spoločné vlastnosti príbehov patrí aj poradie v akom sa udalosti v nich vyskytujú. Francúzsky literárny teoretik Gérard Genette vo svojej knihe Dejová rozprava: Štúdie metód [5] opísal v akom poradí udalosti v deji môžu nasledovať. Práve poradie týchto udalostí môže osobu, ktorej sú podané, ovplyvňovať a tak utvárať jej obraz na celkovú situáciu.

3.2 Programátorské prístupy

Pri vytváraní programu pre generovanie príbehov sa implementory systému zvyčajne opierajú o literárne analýzy diel. Pri takomto prístupe je potrebné vytvoriť model sveta, v ktorom sa príbeh odohráva, umiestniť doň postavy a objekty, určiť medzi nimi vzťahy a priradiť im akcie, ktoré môžu vykonávať. Stretávame sa aj s menej pracnejšími prístupmi, kde program vytvorí príbeh na základe vlastnej analýzy niekoľkých diel. Kvôli veľkému počtu typov diel sú tieto programy špecializované zvyčajne na jeden z nich.

Programy, ktoré generujú príbehy, by mali spĺňať dve základné vlastnosti, a to:

- vedieť generovať udalosti a akcie v určitom poradí
- preformulovať vytvorený príbeh do zrozumitelnejšej formy resp. reči

Na tvorbu takýchto programov sa nám ponúka niekoľko možností. My si ukážeme dve z nich a budeme sa snažiť použiť techniky, ktoré sme si ukázali v podkapitole 3.1.

3.2.1 Generovanie pomocou plánovania

Ak chceme riešiť problém pomocou plánovania, je potrebné, aby sme presne vedeli popísať cieľ, ktorý chceme dosiahnúť, počiatočný stav, v ktorom sa nachádzame, množinu akcií, ktoré môžme použiť a funkciu, ktorá nám povie, aký stav dosianhneme z momentálneho stavu použitím danej akcie. Ak sa pozrieme do podkapitoly 3.1 na Freytagovu analýzu zistíme, že nie je najlepšou voľbou pre plánovanie, pretože neurčuje presné ciele a ani akcie. Preto by sme museli okrem nájdenia daného plánu určiť, či dané poradie akcií spĺňa podmienky stúpania a klesania napätia a to buď po nájdení plánu alebo počas jeho hľadania určitými obmedzeniami na vykonávatelnosť akcií. Oproti tomu Proppove funkcie hovoria presne o cieľoch a akciách ktoré majú byť vykonané, preto je jednoduché ich zapísať ako plánovací problém. Samozrejme je ešte potrebné vytvoriť doménu nad ktorou budú plány pracovať. Výhodou tohoto prístupu je jeho jednoduchosť pri použití správnych postupov a nástrojov.

3.2.2 Generovanie pomocou multiagentového systému

Generovanie príbehu pomocou multiagentového systému bez použitia plánovania je náročnejší proces. Jednotlivých hercov reprezentujeme ako agentov, ktorí majú svoju databázu znalostí a majú splniť zadané ciele. Okrem toho im treba vytvoriť priestor s danými zákonitosťami v ktorom budú existovať.

Existuje viacero multiagentových systémov, ktoré sa opierajú o Proppove funkcie. Pri použití Freytagovej analýzy v multiagentovom programe musíme zabezpečiť nečakané konanie agentov. Nech agenti majú svoju vlastnú neúplnú informáciu o svete. Nasledujúci príklad predstavuje sled udalostí, ktoré môžeme považovať za nečakané.

Drak uniesol princeznu a statočný rytier s výpravou sa ju vydal zachrániť. Po dlhej ceste prišli na miesto, kde drak veznil princeznu. Rytier zasadil mečom silný úder drakovi, avšak meč sa zlomil a tak bol rytier nútený utiecť z boja.

Práve to, že rytier nevedel, že na draka potrebuje silnejšiu zbraň spôsobilo nájdenie akcií ktoré ho priviedli do krajnej situácie. Takýto agent by po vykonaní akcií musel aktualizovať svoje podciele, ktoré sú potrebné na splnenie hlavného cieľa. Predstavené chovanie agentov nevieme všeobecne reprezentovať pomocou plánovacieho problému. Nevýhodou multiagentových systémov je ich zložitosť a veľkosť oproti plánovaniu.

3.2.3 Ďalšie možnosti generovania

Existujú aj netradičnejšie prístupy k tvorbe algoritmov generujúcich príbehy, ktoré sa nezakladajú na literárnych analýzach. Jeden z nich vytvára príbeh na základe modelu, v ktorom herci vyberajú akcie a udalosti, ktoré môžu nastať s určitou pravdepodobnosťou. Pretože tu nie sú presne určené ciele, musia sa dodefinovať niektoré pravidlá, ktoré zaručia vytvorenie príbehu.

Ďalšie algoritmy zas generujú príbehy na základe vlastnej analýzy. Ich vstupom je množina literárnych diel, ktorá sa preskúma a vytvorí sa postup, ktorého by sa mal príbeh držať. Takéto algoritmy nevyžadujú implementáciu sveta a objektov, preto sú menej pracnejšie. Kvalita príbehu však závisí na vstupe a taktiež by diela nemali mať veľmi odlišnú štruktúru.

Pokiaľ sa algoritmy nedržia udalostí, ktoré majú za sebou nasledovať v určitom poradí, potom možu meniť toto poradie. Napríklad ak sú tri udalosti zoradené v chronologickom poradí t.j. v poradí v akom sa za sebou odohrali, možno za určitých okolností prvú a druhú udalosť vymeniť. Táto zámena spôsobí, že pôvodne prvá udalosť bude retrospektíva a príbeh sa začne neobjasnenou udalosťou. Takéto algoritmy používajú systémy ktoré tvoria sujet. Veľa systémov, ktoré sa neopierajú o literárne analýzy obsahujú algoritmy na menenie poradia udalostí.

Kapitola 4

Implementácia

V tejto kapitole popíšeme postupy a prostriedky, ktoré sme použili pri tvorbe nášho systému.

4.1 Príprava

Pred tým ako začneme implementovať systém je vhodné rozmyslieť si, aké literárne konštrukcie budeme chcieť vytvárať a o čo sa budeme pri ich tvorbe opierať. Keďže používame plánovanie a princípy logického programovania, je výhodné použiť Proppovu analýzu, ktorá konštruuje fabulu. Proppov model pre tvorbu príbehu je založený na nasledujúcich požiadavkách:

1. Každý príbeh je zložený z jediného reťazca akcií alebo udalostí ktoré voláme funkcie.

2. Funkcia je významná akcia alebo udalosť definovaná na základe jej miesta v deji.

3. Funkcia nie je motív, téma, postava alebo dej. Je to základná jednotka analýzy.

4. Je jedno kto a ako funkciu vykoná. Z pohľadu analýzy, nie konateľ, jeho metódy a motívy sú podstatné, ale čin samotný je dôležitý.

5. Počet funkcií ktoré má rozprávač k dispozícii je tridsaťjedna.
6. S istými výnimkami, majú funkcie určené striktné poradie.
7. Príbeh sa delí na časti ktoré sa skladajú zo zoradených funkcií.
8. Ku každej funkcii existuje viac variantov jej reprezentácie.
9. Rozprávač môže použiť len sedem chrakteristík postáv.
10. Každý príbeh je zložený z rovnakých funkcií, aj keď sa niektoré nemusia vyskytovať v každom z nich.
11. Všetky príbehy zdieľajú rovnakú základnú štruktúru.

Ak každej funkcii priradíme znak, potom príbeh môžme reprezentovať ako reťazec znakov. Propp urobil viac rozborov literárnych diel a zapísal ich ako postupnosť znakov, ale nedefinoval presne ako môžme tieto postupnosti vytvárať. Jediné, čo pri ich tvorbe vieme je to, že nemôžme vynechať funkciu, ktorá je priamym alebo nepriamym dôsledkom niektorej z predchádzajúcich. Preto si zvolíme pevne určenú postupnosť a jedinú vec ktorú budeme meniť, bude konkrétny variant funkcie.

4.2 Reprezentácia pomocou ASP

Pre implementáciu nášho systému sme zvolili jazyk Clingo, ktorý využíva techniky ASP. Clingo v sebe kombinuje vlastnosti Gringo a Clasp, takže zvláda preložiť aj vyriešiť logické programy, pričom výstupom sú všetky stabilné modely daného programu.

Na začiatok musíme vytvoriť doménu, množinu prvkov nad ktorými budú logické programy pracovať. Tu môžeme deklarovať postavy, miesta, objekty a vzťahy medzi nimi. Čím viac prvkov bude doména obsahovať, tým zložitejšie pravidlá môžeme tvoriť a tak môže byť výsledný príbeh detailnejší. Príklad takejto množiny je nasledovný:

```
character(John). character(Jessica). % postavy
hero(John). princess(Jessica). % priradenie charakterov postavám
object(Pistol). % objekty
location(Hotel room). location(Street). % miesta
path(Hotel room, Street). % vzťah vyjadrujúci dostupnosť miest
on(John,Street). on(Jessica,Hotel room). on(Pistol,John).
% vzťahy medzi postavami, objektami a miestami
```

4.2.1 Všeobecná implementácia pravidiel

Pri tvorbe logického programu musíme dbať na to, aby boli pravidlá v ňom bezpečné, tzn. každá premenná ktorú pravidlo obsahuje, musí byť spomenutá v nejakom pozitívnom literály tohoto pravidla.

Fluenty a akcie

```
fluent(fluent_name(X1,X2,...,Xn)) :- type(X1),type(X2),...,type(Xn).
action(action_name(X1,X2,...,Xn)) :- type(X1),type(X2),...,type(Xn).
```

Generátor akcií

```
 occurs(N,A) :- not -occurs(N,A), state(N), action(A).
-occurs(N,B) :- occurs(N,A), state(N), action(A), action(B), A!=B.
```

Počiatočný stav

```
holds(0,fluent_name(X1,X2,...,Xn)) :- type(X1),type(X2),...,type(Xn).
```

Riešenie Frame problému

```
 holds(N+1,F) :- holds(N,F), not -holds(N+1,F), state(N), fluent(F).
-holds(N+1,F) :- -holds(N,F), not holds(N+1,F), state(N), fluent(F).
```

Požiadavky na vykonávatelnosť akcií

```
:- occurs(N,A), holds(N,F1), -holds(N,F2), not holds(N,F3),
   not -holds(N,F4),...
```

Popis efektov akcii

```
holds(N+1,F) :- occurs(N,A).
```

Popis cieľa

`goal :- holds(N,`F_1`), -holds(N,`F_2`), not holds(N,`F_3`), not -holds(N,`F_4`),...`

Konštanta `N` v predikáte `holds`(resp. `occurs`) určuje kedy má platiť fluent(resp. akcia) v ňom spomenutý.

4.2.2 Konkrétna implementácia niektorých pravidiel

Aby sme vedeli ako presne implementovať Proppove funkcie, ukážeme si konkrétny príklad jednej z nich. Zvoľme si jedenástu funkciu, ktorá hovorí o tom, že nedostatok alebo nešťastie, ktoré spôsobila záporná postava je dané na známosť. Napríklad, ak drak uniesol princeznu a hrdina to nevidel, potom mu niekto musí oznámiť čo sa stalo.

Fluenty

```
fluent(actorAt(A,L)) :- actor(A), location(L).
fluent(captured(A1,A2)) :- actor(A1), actor(A2).
fluent(saw(Actor,Action)) :- actor(Actor), action(Action).
fluent(have(A,O)) :- actor(A), object(O).
fluent(dead(A)) :- actor(A).
```

Akcie

```
action(move(A,L)) :- actor(A), location(L).
action(kidnap(A1,A2)) :- actor(A1), actor(A2).
saction(say(A1,A2,Action)) :- actor(A1), actor(A2), action(Action).
```

Špeciálnu akciu `say` sme implementovali pre potreby rozprávania postáv o tom čo videli. Túto akciu nemôžme klasifikovať pod normálne akcie kôli zacykleniu. Pre špeciálne akcie musíme rozšíriť generátor akcií.

Generátor akcií

```
 occurs(N,A) :- not -occurs(N,A), state(N), action(A).
-occurs(N,B) :- occurs(N,A), state(N), action(A), action(B), A!=B.
-occurs(N,B) :- occurs(N,A), state(N), sAction(A), action(B), A!=B.
 occurs(N,A) :- not -occurs(N,A), state(N), saction(A).
-occurs(N,B) :- occurs(N,A), state(N), action(A), saction(B), A!=B.
-occurs(N,B) :- occurs(N,A), state(N), sAction(A), saction(B), A!=B.
```

Riešenie pre Frame problém je rovnaké ako v 4.2.1.

Požiadavky na vykonávatelnosť akcií

```
:- occurs(N,move(A,L1)), holds(N,actorAt(A,L2)), not path(L2,L1),
   not path(L1,L2), L2!=L1.
:- occurs(N,move(A1,L1)), holds(N,captured(A2,A1)).
:- occurs(N,move(A,L1)), holds(N,dead(A)).
:- occurs(N,say(A1,A2,Act)), holds(N,actorAt(A1,L1)),
   holds(N,actorAt(A2,L2)), L1!=L2.
:- occurs(N,say(A1,A2,Act)), not holds(N, saw(A1, Act)).
:- occurs(N,say(A1,A2,Act)), holds(N,dead(A1)).
:- occurs(N,say(A1,A2,Act)), holds(N,dead(A2)).
:- occurs(N, kidnap(A1,A2)).
```

Popis efektov akcii

```
-holds(N+1,actorAt(A,L)) :- occurs(N,move(A,L1)), holds(N,actorAt(A,L)).
 holds(N+1,actorAt(A,L1)) :- occurs(N,move(A,L1)), holds(N,actorAt(A,L)).
 holds(N+1,saw(A2, Act)) :- occurs(N,say(A1,A2,Act)).
```

Popis cieľa

```
goal :-  holds(n,actorAt(A1,L2)), holds(n,actorAt(A2,L2)),
         holds(n,saw(A2,Act)), holds(n,saw(A1,Act)), hero(A1),
         dispatcher(A2), on(A2,L2).
:- not goal.
```

Zákaz `:- not goal.` zabezpečí odstránenie všetkých modelov, ktoré neobsahujú fluenty uvedené v pravidle `goal`.

4.2.3 Implementačné detaily systému

Pri hľadaní stabilného modelu pre logický program v časti 4.2.2 je potrebné určiť počet akcií ktoré sa majú vykonať. Teraz si musíme položiť otázku, ako zvoliť vhodne konštantu ktorá bude určovať dĺžku plánu? Aj keď určíme túto konštantu dostatočne veľkú, nemusí nám to zaručiť existenciu plánu s danou veľkosťou a naviac ak aj existuje taký plán, je možné že nebude spĺňať akési kritériá dobrého príbehu. Nasledujúci príklad je plán pre prvú z Proppových funkcií, ktorej cieľom je absencia členov určitej skupiny:

```
occurs(0,move(monk,castle))
occurs(1,move(king,forest))
occurs(2,take(princess,blackberry))
```

Je pravda, že sa kráľ v cieľovom stave nenachádza na hrade, avšak v tomto pláne sa vykonali tri akcie ktoré na seba nijako nenadväzujú a zároveň si môžeme všimnúť, že existuje aj kratší plán. Preto je výhodnejšie hľadať najkratšie plány. Konštantu pre dĺžku plánu v Clingu zapíšeme `#const n = x` kde `x` je ohraničené kladné celé číslo.

Ak máme určenú dĺžku plánu môžme začať s jeho hľadaním. Tento proces je však časovo a pamäťovo náročný, preto by sme ho chceli urýchliť. Ak sa pozrieme bližšie na funkcie ktoré sú základnou jednotkou príbehu, zistíme, že niektoré typy akcií ktoré nastanú v jednej funkcii sa nikdy neobjavia v inej. Implementujme každú funkciu samostatne a to tak, že ani jednej neurčíme dĺžku hľadaného plánu a ani jednej okrem prvej z funkcií neurčíme počiatočný stav. Nad takouto množinou logických programov bude operovať nasledovný algoritmus:

```
1: Goal ← ∅
2: LPS ← zoznam logických programov
3: procedure FIND-GOAL
4:     for F in LPS do
5:         if F nie je prvá v LPS then
6:             počiatočný stav F, Init = Goal
7:             Goal ← ∅
8:         end if
9:         while Goal = ∅ do
10:            dĺžka plánu F, n = n + 1
11:            vyrieš F
12:            Goal = množina všetkých fluentov ktoré platia v stave S_n
13:        end while
14:    end for
15: end procedure
```

Priradením cieľového stavu programu P_{n-1} počiatočnému stavu programu P_n zaručíme nadväznosť príbehu. Musíme však zabezpečiť, aby z každého cieľového stavu programu P_{n-1} existoval plán pre program P_n.

Všimnime si, že výstupom logického programu sú jeho všetky stabilné modely. Táto skutočnosť nám rieši otázku interakcie pretože si môžeme vyberať spomedzi viacerých modelov. Zároveň platí, že vybratie ľubovolného modelu nám pri danej implenmentácii globálne nezmení príbeh pretože každý stabilný model je riešením danej funkcie. To znaméná, že všetky stabilné modely musia spĺňať cieľovú podmienku.

4.2.4 Generovanie lepších príbehov

V časti 4.2.3 sme si ukázali ako vytvoriť systém pre generovanie príbehov na základe Proppových funkcií. Takto generované príbehy však nie sú veľmi dlhé a skôr pripomínajú kostru príbehu ktorú ešte možno vylepšiť. Zanedbajme teda časovú a pamäťovú zložitosť, implementujme všetky funkcie v jednom logickom programe a definujme pravidlá pre generovanie, ktoré budú hodnotiť príbeh a zároveň ho umožnia rozšíriť.

Pravidlo 1. *Ak herec alebo dvaja herci vykonajú dve rovnaké akcie pričom výsledkom druhej akcie je stav S_{n+2} ktorého fluenty sú rovnaké ako fluenty stavu S_n, ktorý prislúchal situácii ešte pred vykonaním prvej akcie, ohodnotenie sa zníži pretože sa dej vrátil zo stavu S_{n+1} do stavu S_n.*

Príklad.

$$\begin{aligned} \textit{initially} &\rightarrow S_0 = \{\textit{actorAt(A,}L_1\textit{)}\} \\ \textit{A move to } L_2 &\rightarrow S_1 = \{\textit{actorAt(A,}L_2\textit{)}\} \\ \textit{A move to } L_1 &\rightarrow S_2 = \{\textit{actorAt(A,}L_1\textit{)}\} \end{aligned}$$

Pravidlo 2. *Ak herec vykoná dve rôzne akcie alebo dvaja herci vykonajú dve akcie, pričom akcia jedného z dvoch hercov ovplyvní druhého herca, ohodnotenie sa zvýši, pretože sa dej rozvíja a tiež sa udržiava pozornosť na hercovi.*

Príklad.

initially	$\rightarrow S_0 = \{actorAt(A,L_1)\}$
A move to L_2	$\rightarrow S_1 = \{actorAt(A,L_2)\}$
A take O	$\rightarrow S_2 = \{actorAt(A,L_2),\ have(A,O)\}$

Pravidlo 3. *Ak herec vykoná akciu, ktorá nejakým spôsobom obmedzí druhého herca pri plnení jeho cieľa, ohodnotenie sa zvýši, pretože sa zapletie dej.*

Príklad.

goal	$\rightarrow G = \{actorAt(A_1, L_4)\}$
constrain	$\rightarrow C = \{\leftarrow move(A,L),\ cursed(A).\ \ldots\}$
initially	$\rightarrow S_0 = \{actorAt(A_1, L_1),\ actorAt(A_2, L_2),\ actorAt(A_3, L_3)\}$
A_2 *move to* L_1	$\rightarrow S_1 = \{actorAt(A_1, L_1),\ actorAt(A_2, L_1),\ actorAt(A_3, L_3)\}$
A_2 *curse* A_1	$\rightarrow S_2 = \{actorAt(A_1, L_1),\ actorAt(A_2, L_1),\ actorAt(A_3, L_3),\ cursed(A_1)\}$
A_3 *move to* L_1	$\rightarrow S_3 = \{actorAt(A_1, L_1),\ actorAt(A_2, L_1),\ actorAt(A_3, L_1),\ cursed(A_1)\}$
A_3 *cure* A_1	$\rightarrow S_5 = \{actorAt(A_1, L_1),\ actorAt(A_2, L_1),\ actorAt(A_3, L_3),\ \text{-}cursed(A_1)\}$
A_1 *move to* L_4	$\rightarrow S_6 = \{actorAt(A_1, L_4),\ actorAt(A_2, L_1),\ actorAt(A_3, L_3),\ \text{-}cursed(A_1)\}$

Pravidlá 1 a 2 zabezpečujú "uveriteľnejšie" správanie postáv v príbehu zatiaľ čo pravidlo 3 rozpráva skôr o dramatickosti deja. Implementáciu pravidiel možno nájsť na priloženom CD.

Kapitola 5

Výsledky

V tejto kapitole si ukážeme výsledky implemetovaného systému a zároveň si ukážeme ich rozbor na funkcie pomocou Proppovej analýzy. Ukážeme si aj výsledky systému s implemetovanými pravidlami, ktoré sme popísali v podkapitole 4.2.4.

5.1 Ukážky generované prvým systémom

Príbeh, ktorý si teraz ukážeme je reprezentovaný nasledovnou postupnosťou znakov : $\gamma\beta\delta AB \uparrow DEFGHIJK \downarrow LQW$. Popis Proppových funkcií možno nájsť v prílohe Proppove funkcie.

Ukážka 1.
King says to princess that he want from her to stay at castle.$^{\gamma}$ *King moves to forest.*$^{\beta}$ *Princess moves to forest.*$^{\delta}$ *Ogre kidnaps princess.*^{A} *King moves to castle. King says to blacksmith that ogre kidnapped princess.*^{B} *Blacksmith moves to cloister.*$^{\uparrow}$ *Monk says to blacksmith that he want from him blackberry.*^{D} *Blacksmith moves to castle. Blacksmith moves to forest. Blacksmith takes blackberry. Blacksmith moves to castle. Blacksmith moves to cloister. Blacksmith gives to monk blackberry.*^{E} *Monk gives to blacksmith hammer.*^{F} *Blacksmith moves to castle. Blacksmith moves to forest.*^{G} *Blacksmith fights with*

ogre.^{H} Ogre marks blacksmith.^{I} Blacksmith defeats ogre.$^{J+K}$ Blacksmith moves to castle.$^{\downarrow}$ Princ says to king that princ defeated ogre. Princess moves to castle. Princess says to king that blacksmith defeated ogre.^{L} Blacksmith shows king mark.^{Q} Blacksmith marries princess.^{W}

Ukážka 2.

Queen dies. Monk buries queen in castle. King says to blacksmith that he want from him to stay at castle. Blacksmith moves to cloister. Ogre moves to castle. Ogre kidnaps princess. Blacksmith moves to castle. King says to blacksmith that ogre kidnapped princess. Monk says to blacksmith that he want from him blackberry. Blacksmith moves to forest. Blacksmith takes blackberry. Blacksmith moves to castle. Blacksmith gives to monk blackberry. Monk gives to blacksmith hammer. Blacksmith moves to forest. Blacksmith fights with ogre. Ogre marks blacksmith. Blacksmith defeats ogre. Princess moves to castle. Blacksmith moves to castle. Princ says to king that princ defeated ogre. Blacksmith says to king that blacksmith defeated ogre. Blacksmith shows king mark. Blacksmith marries princess.

Generovanie príbehov sa uskutočnilo pomocou Java programu ktorý spúštal jednotlivé logické programy v Clingo verzie 4.5.0. V nasledujúcej tabuľke sú zaznamenané priemerné časy, potrebné na riešenie jednotlivých funkcií ktoré boli namerané na laptope s procesorom Intel Core I5-2410M o výkone 2.30GHz a RAM o veľkosti 4 GB.

Funkcia	Čas(sec.)	Funkcia	Čas(sec.)	Funkcia	Čas(sec.)
γ	52.145	D	133.25	J+K	0.126
β	0.141	E	2.782	$\downarrow$	0.118
δ	0.118	F	0.217	L	150.109
A	0.127	G	0.141	Q	0.063
B	99.359	H	0.92	W	0.031
$\uparrow$	0.118	I	0.109		

Z tabuľky možno vidieť, že funkcie ktoré potrebovali generátor akcií aký sme si popísali v 4.2.2, potrebovali aj najväčší čas na riešenie.

5.2 Ukážky generované druhým systémom

V tejto podkapitole si ukážeme príbeh, ktorý bol generovaný systémom obohateným o pravidlá z podkapitoly 4.2.4.

Ukážka 3.
King says to blacksmith that he want from him to stay at castle. Blacksmith moves to forest. Blacksmith takes blackberry. Ogre moves to castle. Ogre kidnaps princess. Blacksmith moves to castle. King says to blacksmith that ogre kidnapped princess. Blacksmith moves to cloister. Monk says to blacksmith that he want from him blackberry. Blacksmith gives to monk blackberry. Monk gives to blacksmith hammer. Ogre moves to cave. Ogre takes sword. Ogre moves to forest. Blacksmith moves to castle. Blacksmith moves to forest. Blacksmith fights with ogre. Ogre attacks blacksmith with sword. Blacksmith dodges ogre sword. Dragon marks blacksmith. Blacksmith attack ogre with hammer. Blacksmith hits ogre with hammer. Blacksmith defeats ogre. Princ moves to forest. Princ curse blacksmith. Princ moves to castle. Queen moves to forest. Blacksmith gives to queen flower. Queen cure blacksmith. Blacksmith moves to castle. Princ says to king that princ defeated ogre. Princess moves to castle. Princess says to king that blacksmith defeated ogre. Blacksmith shows king mark. Blacksmith marries princess.

Rozoberme si ukážku 3 a ukážme si ako sa na nej aplikovali pravidlá z 4.2.4. Vykonanie dvoch rôznych akcií si môžeme všimnúť hneď na začiatku kedy kováč odchádza do lesa a pozbiera černice. Takéto chovanie je viac "uveriteľnejšie" oproti ukážke 1 pretože druhá akcia zdôvodňuje, prečo išiel kováč do lesa. Akcie, kedy drak odišiel do jaskyne, aby si zobral meč, spôsobili v konečnom dôsledku, že drak mohol útočiť na kováča a tak stúplo napätie. Podobnú aplikáciu tretieho pravidla si môžeme všimnúť neskôr, kedy princ preklial kováča ktorého následne vyliečila kráľovná. Na koniec musíme podotknúť že sa ukážka 3 taktiež riadi pomocou Proppových funkcií.

Záver

Cieľom tejto bakalárskej práce bolo zostrojiť systém, ktorý genreruje príbehy. Z mnohých prístupov, ktoré sú nám dostupné sme zvolili Proppovu analýzu, ktorú možno nazvať poloformalizmom, a snažili sme sa ju zapísať pomocou ASP. Okrem toho sme definovali pravidlá, ktorých aplikáciou možno generovať ešte lepšie príbehy.

Ďalej sa táto práca zaoberala problematikou interakcie. Ukázali sme si že sa treba pozerať na literárne dielo z rôznych strán, a že je to štruktúra, ktorá má svoje vlastnosti pričom pri ich nesprávnom použití spolu s interakciou môžu pôsobiť protichodne. Okrem iného sme si ukázali aj niektoré postupy tvorby literárnich diel, ktoré môžu byť výhodnejšie pre človeka, ale ťažšie ich formalizovať.

Výpočtová naratológia prešla nemalý kus cesty od jej prvotných aplikácií, avšak stále existuje mnoho vecí, ktoré sú neprebádané alebo ich úplne nechápeme. Jej záber nesmie byť orientovaný len na určité časti literárnej tvorby, pretože niektoré jej aspekty, ako sú napríklad básnické vyjadrovacie prostriedky, sú dôležitou súčasťou literárnych diel.

Literatúra

[1] H. Porter Abbott, Jan Alber, and Marc Alexander. The living handbook of narratology, 2009.

[2] Selmer Bringsjord and David Ferrucci. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press, 1999.

[3] Yundong Cai, Zhiqi Shen, and Chunyan Miao. Agent-oriented methodology for interactive storytelling (aomis). In *Proceedings of the 4th International Conference on Interactive Digital Storytelling*, pages 25–30, Berlin, Heidelberg, 2011. Springer-Verlag.

[4] Tinsley Azariah Galyean III. *Narrative guidance of interactivity*. PhD thesis, Massachusetts Institute of Technology, 1995.

[5] Gérard Genette. *Narrative Discourse: An Essay in Method*. NY: Cornell University Press, 1980.

[6] Pablo Gervás. An expert system for the composition of formal spanish poetry. *Knowledge-Based Systems*, 14(3):181–188, 2001.

[7] Mark Alan Jones. Generating a specific class of metaphors. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 321–323. Association for Computational Linguistics, 1992.

[8] Sheldon Klein, Jon F. Aeschlimann, David F. Balsinger, Steven L. Converse, Claudie Court, Mark Foster, Robin Lao, John D. Oakley, and Joel Smith. Automatic novel writing. Technical report, The University of Wisconsin, 1973.

[9] Michael Lebowitz. Creating a story-telling universe. 1983.

[10] Stefanie Lethbridge and Jarmila Mildorf. Freytag's pyramid. In *Basics of English Studies: An introductory course for students of literary studies in English*. Universities of Tübingen, 2004.

[11] Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.*, 2013.

[12] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1):39–54, 2002.

[13] Brian Magerko and John Laird. Building an interactive drama architecture. In *TIDSE*, 2003.

[14] James H Martin. A computational theory of metaphor. Technical report, DTIC Document, 1988.

[15] James R Meehan. Tale-spin. In *5th International Joint Conferences on Artificial Intelligence*, 1977.

[16] Nick Montfort. *Generating Narrative Variation in Interactive Fiction.* PhD thesis, Philadelphia, PA, USA, 2007.

[17] Rafael Pérez y Pérez. *MEXICA: a computer model of creativity in writing.* PhD thesis, University of Sussex, 1999.

[18] Vladimir Propp. *Morphology of the Folktale.* The American Folklore Society and Indiana University, 1968. Anglický preklad.

[19] Mark O Riedl and Robert Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268, 2010.

[20] Graeme Ritchie. The jape riddle generator: technical specification. *Institute for Communicating and Collaborative Systems*, 2003.

[21] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[22] Eric Steinhart. Netmet: a program for generating and interpreting metaphors. *Computers and the Humanities*, 28(6):383–392, 1994.

[23] Oliviero Stock and Carlo Strapparava. Laughing with hahacronym, a computational humor system. 2006.

Prílohy

Proppove funkcie

Postavy

Vladimir Propp definoval sedem charakteristík postáv ktoré by mali v diele vystupovať.

Hrdina je hlavná postava ktorá vedie príbeh. Obvykle niečo hľadá alebo plní úlohu ktorá mu bola udelená.

Zlosyn bojuje proti hrdinovi. Je morálne zlý a tým vyvoláva myšlienku podpory hrdinu u čitateľa.

Darca poskytne hrdinovi niečo, čo mu v budúcnosti pomôže.

Pomocník podporuje hrdinu. Vystupuje v kritických situáciách. Zvýrazňuje niektoré vlastnosti hrdinu.

Princezna a jej otec je odmena pre hrdinu. Nemusí často vystupovať v príbehu ale môže to byť dôležitá postava.

Dispečer dáva hrdinovi úlohu. Môže byť kombinovaný s inou charakteristikou.

Falošný hrdina sa snaží vystupovať ako hrdina. Môže byť mylne zamenený za hrdinu.

Funkcie

1. **Absencia(β):** Príslušník rodiny odíde z domova. Chýbajúca osoba môže byť dieťa alebo člen staršej generácie. Absenciu osoby môže reprezentovať aj smrť blízkych.

2. **Zákaz(γ):** Hrdinovi sa niečo zakáže alebo je pred niečím varovaný. Taktiež môže dostať rozkaz ktorý musí splniť.

3. **Porušenie zákazu(δ):** Hrdina poruší zákaz. Táto funkcia dopĺňa funkciu 2 a v určitých prípadoch môže existovať aj bez nej.

4. **Výzvedy(ϵ):** Zlosyn hľadá informácie a pokladá otázky obetiam. Táto funkcia existuje aj v inverznej podobe keď sa obeť pýta zlosyna.

5. **Dodanie(ζ):** Zlosyn dostáva priamu odpoveď na svoju otázku.

6. **Klam(η):** Zlosyn sa snaží oklamať obeť aby sa zmocnil kontroly nad ňou. Obeť môže priamo presvedčiť, použiť kúzlo alebo poslať niekoho iného aby presvedčil obeť za neho.

7. **Komplikácia(θ):** Obeť podlieha podvodu a pomáha zlosynovi.

8. **Podlosť, potreba(A):** Členovi rodiny ublíži zlosyn alebo mu niečo chýba a túži po tom.

9. **Rozjímanie(B):** Hrdinovi je uložená požiadavka. V tejto funkcii sa uvedie hrdina a je vyslaný splniť úlohu.

10. **Protiakcia(C):** Hrdina sa rozhodne konať a prijme úlohu.

11. **Odchod($\uparrow$):** Hrdina opúšťa domov. V tejto funkcii je v príbehu uvedený darca.

12. **Prvá funkcia darcu(D):** Darca môže testovať hrdinu, vypočúvať ho alebo ho o niečo žiadať.

13. **Reakcia hrdinu(E):** Hrdina reaguje na darcove požiadavky a to kladne alebo záporne.
14. **Získanie magického agenta(F):** Hrdina získa magický predmet, zviera alebo osobu ktorý mu v budúcnosti pomôže.
15. **Premiestnenie(G):** Hrdina sa premiestni k objektu ktorý hľadá.
16. **Zápas(H):** Hrdina bojuje so zlosynom. Boj nemusí byť len fyzický ale môže to byť aj hra.
17. **Označenie(I):** Hrdina je poznačený v boji. Tiež môže získať predmet ktorý dokazuje, že sa boj odohral.
18. **Víťazstvo(J):** Zlosyn prehráva boj s hrdinom.
19. **Vyrovnanie(K):** Hrdina nájde čo hľadal a tak splní úlohu ktorá mu bola udelená.
20. **Návrat(↓):** Hrdina sa vydá na cestu domov.
21. **Stíhanie(Pr):** Hrdina je prenasledovaný alebo je otrávený jedom poprípade zranený z boja.
22. **Záchrana(Rs):** Hrdina je zachránený pred prenasledovateľom resp. vyliečený.
23. **Nebadaný príchod(O):** Hrdina prichádza nepozorovane domov. Môže sa stať že ho nikto nespozná.
24. **Neoprávnené požiadavky(L):** Falošný hrdina sa dožaduje odmeny.
25. **Ťažká úloha(M):** Na hrdinu je kladená ťažká úloha.
26. **Vyriešenie(N):** Hrdina splní úlohu.
27. **Rozpoznanie(Q):** Hrdina je spoznaný podľa značky ktorú si priniesol z boja.

28. **Odhalenie(Ex):** Falošný hrdina alebo zlosyn je odhalený.
29. **Premena(T):** Zmení sa vzhľad hrdinu alebo jeho postavenie.
30. **Trest(U):** Zlosyn je potrestaný.
31. **Svadba(W):** Hrdina sa ožení, získa kráľovskú korunu alebo je odmenený.