VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# SIMULACE VYJEDNÁVACÍCH A ARGUMENTAČNÍCH PROTOKOLŮ

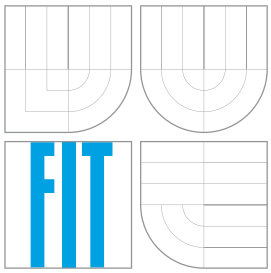
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL ŘÍHA

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# SIMULACE VYJEDNÁVACÍCH A ARGUMENTAČNÍCH PROTOKOLŮ

SIMULATION OF NEGOTIATION AND ARGUMENTATION PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL ŘÍHA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FRANTIŠEK ZBOŘIL, Ph.D.

BRNO 2010

# Abstrakt

Tato práce se zabývá komunikací v multiagentních systémech, a to konkrétně vyjednáváním a argumentací. Jsou popsány protokoly pro argumentaci a vyjednávání, a je uveden modelový příklad jejich využití. Je popsán hierarchický model důvěry v kontextu, který slouží agentům k reprezentaci jejich představ o systému. Pro tyto agenty je pak navržen argumentační protokol, kterým jsou řešeny konflikty.



# Abstract

This work deals with communication in multiagent systems. The protocols for negotiation and argumentation are shown, and model example of their usage is described. We describe hierarchical model of trust in contexts, that is used for representation of agent's believes. The argumentation protocol for those agents is designed, and is used for solving conflicts.



# Klíčová slova

agenti, vyjednávání, argumentace, hierarchický model důvěry v kontextu

# Keywords

agents, negotiation, argumentation, hierarchical model of trust in contexts


# Citace

Michal Říha: Simulace vyjednávacích a argumentačních protokolů, diplomová práce, Brno, FIT VUT v Brně, 2010

# Simulace vyjednávacích a argumentačních protokolů

## Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Františka Zbořila, Ph. D. Uvedl jsem všechny zdroje ze kterých jsem čerpal.

.....
Michal Říha
23. května 2010

## Poděkování

Děkuji panu Ing. Františku Zbořilovi, Ph.D. a panu Ing. Janu Samkovi za spolupráci při tvorbě této práce.

# Obsah


**1 Úvod** **3**

**2 Agenti a agentní systémy** **4**
2.1 Systém . . . . . 4
2.2 Umělý agent . . . . . 5
2.3 Multiagentní systém . . . . . 5
2.3.1 Prostředí bez komunikace . . . . . 6
2.3.2 Prostředí s komunikací . . . . . 6

**3 Vyjednávací a argumentační protokoly** **7**
3.1 Vyjednávání . . . . . 7
3.1.1 Vyjednávací protokol . . . . . 8
3.2 Argumentace . . . . . 9
3.2.1 Formální systém argumentace . . . . . 9
3.2.2 Argumentační protokol . . . . . 10
3.2.3 Postoje agentů . . . . . 11

**4 Model agentního systému s vyjednáváním** **12**
4.1 Popis modelu . . . . . 12
4.2 Popis komunikace . . . . . 12
4.3 Příklady vyjednávaných problémů . . . . . 13

**5 Hierarchický model důvěry v kontextu** **15**
5.1 Neformální popis . . . . . 15
5.2 Struktura modelu . . . . . 16
5.3 Důvěra . . . . . 17
5.4 Výpočet důvěry . . . . . 18
5.4.1 Výpočet up-directon . . . . . 18
5.4.2 Výpočet down-direction . . . . . 19
5.5 Navržený model . . . . . 19

**6 Argumentace** **21**
6.1 Konflikt . . . . . 21
6.2 Události . . . . . 21
6.3 Klíčové uzly . . . . . 22
6.4 Váha důvěry . . . . . 23
6.5 Argument . . . . . 24

**7 Argumentační protokol** **26**
7.1 Popis protokolu . . . 26
7.2 Assert . . . 27
7.3 Challenge . . . 27
7.4 Accept . . . 28
7.5 End, Endconfirm . . . 28
7.6 Příklad komunikace . . . 28

**8 Dosažené výsledky** **31**
8.1 Modelový systém . . . 31
8.2 Příklad . . . 32
8.2.1 Představy agenta A . . . 33
8.2.2 Představy agenta B . . . 34
8.2.3 Stav po argumentaci . . . 34

**9 Možnosti rozšíření** **36**
9.1 Klíčové uzly . . . 36
9.1.1 Dynamické klíčové uzly . . . 36
9.2 Konstrukce argumentů . . . 37
9.3 Popiratelnost události . . . 38
9.4 Odstranění vzniku nových konfliktů . . . 38

**10 Závěr** **39**

# Kapitola 1

# Úvod

V současné době vzniká velký počet systému, využívající autonomně fungující agenty. K jejich nasazení dochází v širokém spektru disciplín. S tím souvisí vzrůstajících počet multiagentních systému. Jako příklad aplikace multiagentního systému můžeme uvést systém Oasis, který měl pomocí racionálních agentů řídit letiště v Sydney.

V multiagentních systémech jsou na agenty kladeny vyšší nároky. Agent se musí chovat sociálně, tedy komunikovat s ostatními agenty. Zároveň by měl být schopen s nimi kooperovat, případně také bojovat o omezené prostředky. V tomto prostředí je významným schopností agenta umění vyjednávat, případně vést spor s jiným agentem.

Tato práce se zabývá návrhem argumentačního protokolu pro konkretní případ. Agenti mají představy o určitém systému, které jsou vyjádřeny pomocí hierarchického modelu důvěry v kontextu. Pomocí navrženého protokolu se pak snaží řešit konfliktní situace v těchto modelech.

V kapitole 2 jsou popsány základní poznatky o agentech a agentních systémech. Kapitola 3 se podrobně zabývá vyjednávacím a argumentačním protokolem. V kapitole 4 je pak nastíněn modelový příklad využití argumentačních a vyjednávacích protokolů. Kapitola 5 popisuje hierarchický model důvěry v kontextu. V kapitole 6 jsou popsány základní části navrženého způsobu argumentace. Kapitola 7 pak popisuje konkretně navržený protokol. V kapitole 8 je popsán příklad funkčnosti protokolu. V kapitole 9 jsou pak shrnuty nedostatky navrženého protokolu.

# Kapitola 2

# Agenti a agentní systémy

Tato kapitola obsahuje definici pojmu jako je systém, umělý agent a multiagentní systém, které jsou důležité pro další kapitoly. Nejprve je definováno, co je to systém. Z této definice je poté odvozen umělý agent a prostředí, ve kterém je situován. Poslední část pak shrnuje tyto poznatky v definici multiagentních systémů.

## 2.1 Systém

Systémem rozumíme množinu prvků, ve které každé dva prvky mají jistý vztah. Pro přesnější zápis je vhodné zavést množinu všech prvků v systému, takzvané universum, které se značí symbolem U, a množinu relací na množině universa, kterou označuje symbol R a která se nazývá charakteristika. Systém S je pak možné vyjádřit zápisem

$$S =< U, R >$$

Každý prvek $u_i \in U$ má vstupní brány $X^{u_i}$, které jsou podmnožinou všech vstupních bran systému $X$. Vstupními branami prvek získává informace o svém okolí. Zároveň má výstupní brány $Y^{u_i}$, které jsou podmnožinou všech výstupních bran systému $Y$. Výstupními branami prvek ovlivňuje své okolí. Propojením vstupních a výstupních bran mezi jednotlivými prvky vznikne vazba, která je prvkem charakteristiky systému. Charakteristiku systému tedy můžeme zapsat jako $R \subseteq XxY$.

Stav systému v určitém čase je dán stavem jeho strukturou a stavem jeho prvků. Stav prvku je dán vnitřním stavem a hodnotami na vstupních a výstupních branách. Stav systému se v čase mění podle toho, jak se mění stav jeho prvků. Jestliže je tato změna dána pouze chováním prvků které systém obsahuje, mluvíme o uzavřeném systému. Na otevřený systém působí kromě jeho prvků ještě také okolí.

Stav prvku $u$ je určen hodnotami vektoru jeho vstupů $\vec{x}$, výstupů $\vec{y}$ a jeho vnitřním stavem $s_u$. Tyto struktury tvoří konfiguraci prvku $c_u =< \vec{x_u}, \vec{y_u}, s_u >$. Stav celého systému je pak dána konfigurací všech jeho prvků a charakteristikou

$$C =< \vec{c_u}, R >$$

Systém v čase postupně přechází z jednoho stavu do jiného. Tomuto průběhu se říká chování systému. Na chování systému má kromě jeho prvků vliv také okolí u otevřených systému, případně systém může být nedeterministický. Chování lze zapsat jako zobrazení

$$\xi : [X]_s^T \rightarrow [Y]_s^T$$

Pro všechny možné posloupnosti hodnot na vstupech prvků daných zobrazení časových okamžiků na vektor vstupního univerza existuje odpovídající posloupnost hodnot na výstupech daných zobrazením z množiny časových okamžiků na vektor výstupního univerza.

## 2.2 Umělý agent

Umělý agent je člověkem vytvořené dílo, které v prostředí, do kterého je umístěno, jedná samostatně v prospěch svého klienta [5]. Agentem tedy rozumíme aktivní prvek v systému, který bez vnějšího řízení dokáže plnit úkoly, pro které je určen. Důležitou podstatou je, že agent se chová samostatně. Takto navržené agenty lze využít v široké oblasti systémů. Agent provádí svou činnost v určitém prostředí, říkáme že agent je situován v danném prostředí.

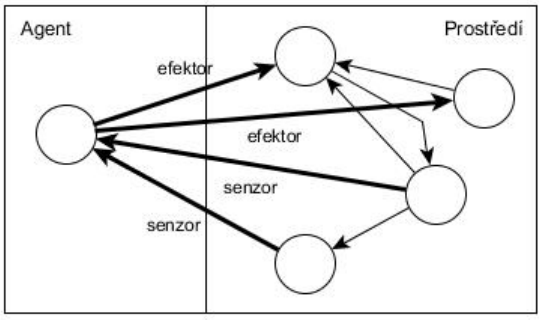
Prostředí ve kterém je agent situován může být statické/dynamické, spojité/nespojité, deterministické/nedeterministické, epizodní případně strategické. Z pohledu této práce je poslední jmenované, strategické prostředí, nejzajímavější, neboť je to prostředí které sdílí více agentů. Jednotlivým agentům se takovéto prostředí může jevit jako nedeterministické a dynamické, i když tomu tak není. Vhodným modelováním prostředí může agent předpokládat vývoj a rozhodovat se tak o svých akcích. Významnou součástí takovéhoto prostředí může být komunikace mezi agenty, o které tato práce pojednává.

Na samotného agenta lze pohlížet jako na systém. Takovýto systém má prvky

$$S_{agt} = (U_{agt}, R_{agt}), \xi_{agt}, T_{agt}, C_{agt}$$

$\xi_{agt}$ je chování agenta, které je definováno pro časovou množinu $T_{agt}$. Agent se může nacházet ve stavech daných množinou $C_{agt}$. Agentní systém umístěný do prostředí ukazuje obrázek 2.1.

Obrázek 2.1: Agentní systém situovaný v prostředí.




## 2.3 Multiagentní systém

Agentní systém je takový, který obsahuje prvek mající funkci být agentem. Tento prvek přijímá podněty ze svého okolí, na jejich základě se rozhoduje a své okolí opětovně ovlivňuje.

Mluvíme-li o multiagentních systémech, pak se v něm vyskytuje více agentů. Multiagentní systém tvoří prostředí a populace s více agenty. Součástí takového prostředí jsou interakce mezi agenty a interakce více agentů s prostředím. Model systému můžeme zapsat jako

$$M =< U, R >$$

Množina $U$ všech prvků systému obsahuje populaci agentů $\Theta$ a množinu prvků v prostředí $E$. Platí tedy $U = \Theta \cup E$ a zároveň $\Theta \cap E = \emptyset$.

Charakteristiku systému $R$ můžeme rozdělit na tři podmnožiny, kde $R_{agt}$ jsou vzájemné vazby mezi agenty, $R_{env}$ jsou vzájemné vazby mezi prvky prostředí a $R_{irc}$ jsou pak vazby mezi agenti a prvky prostředí. Toto rozdělení lze zapsat jako $R = R_{agt} \cup R_{env} \cup R_{irc}$.

### 2.3.1 Prostředí bez komunikace

Předpokládejme racionální agenty, kteří se snaží dosáhnout svých cílů, nemají však možnost mezi sebou komunikovat. Chování těchto agentů v multiagentním prostředí lze popsat pomocí teorie her. Pokud má agent $A$ ze stavu prostředí $e$ užitek $U = (A, e)$, pak agent vždy preferuje $e$ nad $e'$ pokud $U(A, e) > U(A, e')$. Agent volí takovou strategii z množiny možných strategií, která je takzvanou „best response“ na strategie ostatních agentů. Taková strategie má nejvyšší očekávaný užitek. Jestliže existuje strategie, která je vždy nejlepší odpovědí na jakoukoliv strategii ostatních agentů, nazývá se tato strategie dominantní strategií, a agent ji bude volit ve všech situacích.

### 2.3.2 Prostředí s komunikací

Pokud prostředí, ve kterém jsou agenti umístěni umožňuje komunikaci, dochází mezi agenty k interakci. Důvodem k interakci může být požadavek na splnění cíle, pro jehož uskutečnění nemá agent potřebné prostředky. Dalším důvodem k interakci mohou být sdílené prostředky. Takový prostředek může využívat nejvýše jeden agent, a proto dochází v jeho přístupu ke konfliktům. V případě, že jím disponuje agent, který jej sám nevyužívá, může jej poskytnout dalším agentům.

Konflikty v multiagentním systému lze rozdělit do dvou kategorií, a to na konflikty fyzické a mentální. Fyzické konflikty jsou konflikty o prostředí, tedy například o sdílené zdroje, služby, komunikační kanály a podobně. Tyto konflikty se řeší vyjednáváním, popřípadě koordinací agentů v přístupu ke zdrojům. Mentální konflikt je konflikt v představách, závazcích a cílech agentů. Takovýto konflikt může nastat u jednoho agenta. V tom případě musí agent provést takové kroky, aby jeho mentální stav byl konzistentní. Při mentálním konfliktu mezi více agenti je základem řešení komunikace, zejména vyjednávání a argumentace. Při ní se agent snaží prosadit své mentální postoje, které podporuje argumenty.

Pro rešení konfliktů v multiagentních systémech existuje několik protokolů:

- **Dražby** - několik agentů žádá přístup k sdílenému prostředku, o jeho přidělení soutěží v dražbě.
- **Volby** - v případě že nastane situace, pro kterou existuje více řešení, je zvoleno jedno z nich na základě preferencí všech agentů v multiagentním systému.
- **Vyjednávání** - agenti hledají kompromis řešeného konfliktu.
- **Argumentace** - protokol pro řešení mentálních konflitků.

Protokoly vyjednávání a argumentace jsou jádrem této práce, a zabývá se jimi kapitola 3.

# Kapitola 3

# Vyjednávací a argumentační protokoly

Tato kapitola detailně popisuje vyjednávací a argumentační protokoly. Nejprve je popsán princip vyjednávání, který je následně doplněn přesným protokolem. Obsah této sekce vychází ze článku [1]. Sekce 3.2.1 pojednává o argumentaci. Nejprve je definován formalní systém pro argumentaci, který je poté rozšířen o argumentační protokol. Podsekce 3.2.3 pak popisuje možné postoje agentů k předkládaným faktům. Sekce o argumentaci vychází ze článku [2].

## 3.1 Vyjednávání

Vyjednávání je proces který probíhá mezi dvěma nebo více agenty. Cílem vyjednávání je nalézt možný postup k dosažení cílů agenta v případě, že agent není schopen úlohu vyřešit sám, nebo ji z určitých důvodů sám řešit nechce. Cíle vyjednávajících agentů mohou být protichůdné, popřípadě závislé na splnění cílů jiných agentů. Proces vyjednávání vždy obsahuje výměnu nabídek (proposal), kritik (critique), vysvětlení (explanations) a meta-informací.

Nabídka je základním návrhem pro vyřešení určitého problému. Každé vyjednávání začíná v momentě, kdy jeden agent udělá nabídku jinému agentovi. Nabídka může obsahovat návrh na částečné vyřešení problému, kompletní řešení problému, případně protislužbu kterou je agent ochotný vykonat. Příkladem takové nabídky může být:

**A: Navrhuji, že ti poskytnu službu Y v případě, že mi poskytneš službu Z.**

Kritika je jedním ze způsobů, kterým může agent odpovědět na příchozí nabídku. Obsahuje odpověď na původní nabídku, kterou buď přijme, nebo odmítne, případně může informovat o tom, které části původní nabídky agentovi nevyhovují. Proces generování kritiky na nabídku je momentem, kdy agent vyhodnocuje předchozí nabídku s ohledem na své cíle. Specifikováním části nabídky s kterou agent nesouhlasí se snaží dostat jinou, lépe vyhovující nabídku. Příkladem kritiky můžou být následující zprávy

**A: Navrhuji, že mě poskytneš službu X.**
**B: Souhlasím.**

**A: Navrhuji, že ti poskytnu službu Y v případě že mi poskytneš službu X.**
**B: Nemám zájem o službu X.**

Druhým způsobem odpovědi na nabídku je protinabídka. V ní agent převezme část původní nabídky, kterou upraví aby lépe vyhovovala jeho cílům. Příkladem může být například komunikace

**A: Navrhuji, že ti poskytnu službu Y v případě že mi poskytneš službu X.**
**B: Navrhuji, že ti poskytnu službu X v případě že mi poskytneš službu Z.**

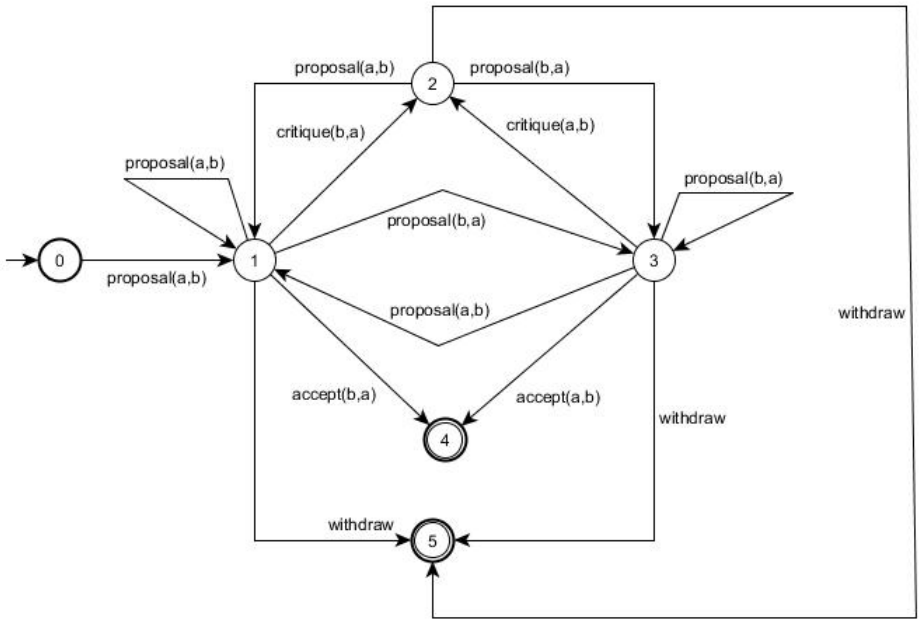
Nabídky, kritiky a protinabídky jsou typy zpráv, které přímo zahrnují co agent chce. Pokud k je k těmto zprávám přiloženo i vysvětleni, lze snadněji dosáhnout dohody. Vysvětlení obsahuje důvod, proč agent požaduje souhlas s nabídkou. Slouží k přesvědčení druhé strany, že argument je platný. Cílem metainformací je pak poskytnout druhému agentovi informace, které je schopen využít pro zmenšení prohledávaného prostoru možných nabídek.

### 3.1.1 Vyjednávací protokol

V případě vyjednávání, kterého se účastní pouze dva agenti je možné vyjednávací protokol popsat stavovým diagramem, který je na obrázku 3.1. Proces začíná zasláním nabídky. $\phi$ v diagramu značí nabídku včetně možného vysvětlení, pokud se jej agent rozhodne poskytnout. V momentě kdy agent zašle nabídku, může zaslat jinou nabídku bez čekání na odpověď na předchozí. Druhý agent může na nabídku reagovat akceptováním nabídky, kritikou nabídky, protnabídkou, nebo může odstoupit od procesu vyjednávání. Pokud agent kritizuje předchozí nabídku, nebo učiní protinabídku, posune se proces do stavu kdy každý agent může zaslat jinou nabídku. Celý proces iteruje do okamžiku, kdy jeden agent nabídku přijme, nebo vyjednávání ukončí.

Z diagramu vyplývá, že protokol nerozlišuje mezi nabídkami a protinabídkami. Akce přijmutí a odmítnutí jsou speciálním druhem kritiky, který vyjednávání ukončuje.

Obrázek 3.1: Stavový diagram vyjednávacího protokolu mezi dvěma agenty.



## 3.2 Argumentace

Argumentace je proces probíhající mezi dvěma agenty. Jeho cílem je řešení mentálních konfliktů, které mohou v multiagentním systému nastat. Agenti se snaží přesvědčit svého oponenta předkládáním faktů, které jsou podloženy argumenty. Cílem je změna mentálního stavu jednoho ze zúčastněných agentů.

### 3.2.1 Formální systém argumentace

Mějme bázi znalostí $\sum$, která může být nekonzistentní. Argument A je poté dvojice $A = (H, h)$, kde h je formule daného jazyka a H je podmnožina $\sum$, pro kterou platí:

1. H je konzistentní
2. H $\vdash$ h
3. H je minimální, čili neexistuje podmnožina H, která by splňovala zároveň podmínky 1 a 2.

H se nazývá podpora argumentu A, zapsáno $H = Support(A)$. h se nazývá závěr, psáno $h = Conclusion(A)$.

K zachycení faktu, že některá fakta mohou být více důvěryhodná než jiná je na množině faktů definována relace preference. Jednotlivým argumentům, jejich třídám nebo formám je přiřazena hodnota, která udává jejich váhu. Tím na bázi znalostí vznikne neostré uspořádání $>>_{Pref}$. Síla argumentu je pak dána silou nejslabší formule vyskytující se v podpoře argumentu.

Vzhledem k tomu, že $\sum$ je nekonzistentní, argumenty které lze na této množině vytvořit budou konfliktní. Všechny argumenty, které lze vytvořit na množine $\sum$ označíme jako $A(\sum)$. Napadání pravdivosti daného argumentu může mít dvě formy, a to vyvracení (rebut) nebo zpochybnění (undercut).

Mějme dva argumenty $A_1$ a $A_2$. $A_1$ poté zpochybňuje $A_2$ v případě, že závěr $A_1$ je negací některé formule, která je podporou $A_2$. Formálně je zpochybnění definováno následovně:

Nechť $A_1$ a $A_2$ jsou dva argumenty z $A(\sum)$. $A_1$ zpochybňuje $A_2$, jestliže $\exists h \in support(A_2)$ takové, že $h \equiv \neg conclusion(A_1)$.

Argumentační systém můžeme definovat následujícím způsobem:

Argumentační systém (AS) je trojice $< A(\sum), Undercut, >>_{Pref} >$, kde

- $A(\sum)$ je množina argumentů, které lze odvodit z $\sum$
- Undercut je binární relace mezi argumenty, značící že jeden argument zpochybňuje druhý. $Undercut \subseteq A(\sum) x A(\sum)$.
- $>>_{Pref}$ je uspořádání na množině $A(\sum)$

Mezi argumenty $A_1$ a $A_2$ z množiny $A(\sum)$ můžeme rozlišit různé druhy vztahů:

Pokud $A_1$ zpochybňuje $A_2$, pak se $A_2$ ubrání proti $A_1$ v případě že $A2 >>_{Pref} A1$. V opačném případě se neubrání.
Množina argumentů ubrání A pokud: $\forall B$ zpochybňující A a pro která se A neubrání proti B platí, že $\exists C \in S$ takové, že C zpochybňuje B a B se neubrání proti C.

$C_{Undercut,Preff}$ je poté množina argumentů, které nejsou zpochybněny nebo které se sami zpochybnění ubrání. Množina přípustných argumentů argumentačního systému $< A(\sum), Undercut, >>_{Pref}>$ je poté pevný bod funkce F:

$F(S) = \{(H,h) \in A(\sum) | (H,h)\ je\ braneno\ S\}$
$S \subseteq A(\sum)$

Množina přípustných argumentů argumentačního systému
$< A(\sum), Undercut, >>_{Pref}>$ je:

$\underline{S} = \bigcup F_{i \geq 0}(\emptyset) = C_{Undercut,>>_{Pref}} \cup [\bigcup F_{i \geq 1}(C_{Undercut,>>_{Pref}})]$

Argument je přípustný pokud je členem přípustné množiny.

### 3.2.2 Argumentační protokol

Mějme dva agenty A a B. Každý agent má bázi znalostí, $\sum A$ a $\sum B$. Agenti jsou schopni určit, pro které nabídky mají akceptovatelné argumenty. Navíc každý agent má další znalosti, které obsahují vyřčené a přijaté formule partnera. Tyto množiny se nazývají commitment store, a označují se CS(A) a CS(B). Na sjednocení těchto množin lze nahlížet jako na stav komunikace v daném čase. Jelikož má každý agent tyto dvě báze znalostí, využívá pak argumentační systém $< A(\sum \cup CS(A)), Undercut, >>_{Pref}>$, respektive
$< A(\sum \cup CS(B)), Undercut, >>_{Pref}>$.

Následující část popisuje jednotlivé atomické akce mezi agenty. Předpokládá se, že zprávu zasílá agent A agentovi B. Zprávy **assert** a **accept** upravují commitment store odesílajícího agenta. Zprávy **challenge** a **question** nemají na commitment store vliv.

**assert**($p$)
$p$ je navrhovaná formule
$CS_i(A) = CS_{i-1}(A) \cup \{p\} \wedge CS(B)_i = CS(B)_{i-1}$
Agent A předává agentovi B informaci o pravdivosti $p$.

**assert**($S$)
$S$ je množina formulí
$CS_i(A) = CS_{i-1}(A) \cup \{S\} \wedge CS(B)_i = CS(B)_{i-1}$
Agent A předává agentovi B množinu argumentů $S$.

**accept**($p$)
$p$ je navrhovaná formule
$CS_i(A) = CS_{i-1}(A) \cup \{p\} \wedge CS(B)_i = CS(B)_{i-1}$
Agent A přijímá pravdivost formule $p$.

**accept**($S$)
$S$ je množina formulí
$CS_i(A) = CS_{i-1}(A) \cup \{S\} \wedge CS(B)_i = CS(B)_{i-1}$
Agent A přijímá pravdivost argumentů v množině $S$.

**challenge($p$)**
$p$ je navrhováná formule
$CS_i(A) = CS_{i-1}(A) \wedge CS(B)_i = CS(B)_{i-1}$
Agent A vyzývá agenta B k explicitnímu vyjádření argumentů, které podporují formuli $p$.

**question($p$)**
$p$ je navrhovaná formule
$CS_i(A) = CS_{i-1}(A) \wedge CS(B)_i = CS(B)_{i-1}$
Agent A se dotazuje agenta B na pravdivost formule $p$.

### 3.2.3 Postoje agentů

S ohledem na popsané atomické akce zůstává otázkou, jakým způsobem agenti navrhují formule pro akce **assert**, a jak snadno přijímají formule akcí **accept**.

Nejprve se budeme zabývat přístupu agentů k akci **assert**. Agent může mít jeden ze dvou následujících přístupů k předkládání formulí:

- **Přesvědčený** (confident) agent předloží formuli $p$, pokud pro ni ve své bázi znalostí může sestrojit argument $(S, p)$.
- **Uvážlivý** (thoughtful) agent předloží formuli $p$, pokud pro ni ve své bázi znalostí může sestrojit přípustný argument $(S, p)$.

Uvážlivý agent předkládá pouze formule, o kterých si myslí, že jsou pravdivé. Přesvědčený agent předkládá jakoukoliv sestrojitelnou formuli, přičemž její pravdivost ve své bázi znalostí neověřuje.

Nyní popíšeme postoje k přijímání formulí akcí **accept**. Agent může mít jeden ze tří následujících přístupů k přijímání formulí:

- **Důvěřivý** (credulous) agent přijme jakoukoliv formuli $p$, pokud je podložena argumentem $(S, p)$.
- **Obezřetný** (cautious) agent přijme formuli $p$, pokud není schopen sestrojit silnější argument pro $\neg p$.
- **Skeptický** (sceptical) agent přijme formuli $p$, pokud pro ni existuje přípustný argument.

Skeptický agent požaduje nejvyšší ověření správnosti před akceptováním dané formule. Typicky na jakoukoliv akci **assert($p$)** odpoví **challenge($p$)**, a obdržený argument podrobí kontrole vzhledem ke své bázi znalostí.

# Kapitola 4

# Model agentního systému s vyjednáváním

Tato kapitola popisuje model multiagentního systému, který je navržen pro využití vyjednávání a argumentace. Model je převzat z článku [4].

## 4.1 Popis modelu

Model uvažuje multiagentní systém pro Britský Telecom, který vyhodnotí cenu za vytvoření sítě, která poskytuje speciální služby pro zákazníka. Proces dostane od zákazníka požadavek, a jako výstup vyhodnotí cenu, kterou bude realizace stát. Do průběhu jsou zapojeni následující agenti: oddělení služby pro zákazníky (customer service division, CSD), návrhové oddělení (design division, DD), inspekce (surveyor department, SD) a agenti kteří poskytují službu prověření zákazníka (vetting agents, VC). VC agenti nejsou přímo součástí společnosti a službu poskytují způsobem outsorcingu.

Proces začíná agent CSD, který zjistí základní údaje o zákazníkovi a prověří jeho schopnost platby. To se děje pomocí VC agentů. Pokud zákazník projde kontrolou, je jeho požadavek porovnán se standardními službami společnosti. V případě, že jeho požadavek je v rámci těchto služeb splnitelný, lze mu ihned poskytnout nabídku. Pokud služba není snadno splnitelná, CSD agent vyjedná s DD agentem službu, která navrhne potřebnou síť a určí cenu. Zároveň DD agent rozhodne, zda by měla proběhnout inspekce stránky. Agent DD vyjednává s agentem SD o této službě. Zvláštností tohoto vyjednávání je, že agent DD má vyšší autoritu než agent SD, čili agent SD nemůže službu odmítnout. Po dokončení veškerých těchto úkonů informuje agent CSD zákazníka o nabídce společnosti na danou službu. Tím se proces ukončí.

## 4.2 Popis komunikace

Agenti spolu komunikují pomocí obecného komunikačního jazyka (communication language, CL) definovaného nad množinou atomických akcí, jejíž obsah je vyjádřen logickým jazykem L. Jazyk L není přesně specifikován, nicméně musí obsahovat alespoň tyto základní prvky:

- Proměnné - reprezentující vyjednávané hodnoty, které se během vyjednávání mění
- Konstanty - reprezentující vyjednávané hodnoty. Speciální konstantou je „?“, který reprezentuje chybějící hodnotu.

- Rovnost
- Konjunkci

Poslední součástí tohoto jazyka musí být relace preference. Příkladem věty takového jazyka může být například:

$$(Cena = 100) \wedge (Kvalita = Vysoka)$$

Komunikační jazyk je množina atomických akci, které jsou rozděleny do dvou množin. První množínou je $I_{nego}$, která odpovídá vyjednávání. Druhá je pak $I_{pers}$, odpovídající argumentaci. Množiny obsahují následující akce

- $I_{nego} = \{offer, request, accept, reject, withdraw\}$
- $I_{pers} = \{appeal, threaten, reward\}$

Vyjednávací dialog mezi dvěma agenty je pak sekvence nabídek a protinabídek, které mohou být doplněny přesvědčovacímy argumenty. Přesvědčováním se agent snaží změnit představy a cíle druhého agenta.

Formát vyjednávacích atomických akcí odpovídá zápisu uvedenému v kapitole 3.1.1. Akce pro argumentaci mají formát následující:

$appeal(A, B, \Phi, (\neg)\varphi, t)$

A - zasílající agent
B - přijímající agent
$\Phi$ - argument
$\varphi$ - podpora argumentu

$reward(A, B, (\neg)\Psi_1, (\neg)\Psi_2, t)$
$threaten(A, B, (\neg)\Psi_1, (\neg)\Psi_2, t)$

A - zasílající agent
B - přijímající agent
$(\neg)\Psi_1$ - akce, která se má provést. Její negace znamená že se akce neprovede.
$(\neg)\Psi_2$ - odměna, případně hrozba která se splní, pokud nastane akce $(\neg)\Psi_1$

## 4.3 Příklady vyjednávaných problémů

Prvním příkladem vyjednávaného problému je požadavek agenta CSD na prověření společnosti A. Požadavek zasílá VC agentovi. Agent CSD požaduje, aby služba byla vyřízena během 24 hodin a je ochoten za ni zaplatit 1000Kč. Agent VC odpoví, že společnost A je známa finančními obtížemi a proto její prověření bude více náročné. Navíc požaduje odložení, neboť již slíbil službu jinému zákazníkovi, označenému jako B. Tento dialog reprezentuje následující sekvence:

1. $offer(CSD, VC, Spolecnost = A \wedge Cena = 1000 \wedge Cas = 24h, t_1)$
2. $appeal(VC, CSD, Spolecnost = A \wedge Cena = 2000 \wedge Cas = 48h,$
   $FinancniStatus = Spatny \wedge SlozitostProvereni = Vysoka, t_2)$

3. $appeal(VC, CSD, Spolecnost = B \wedge Spozdeni = 24h,$
   $accept(VC, CSD, Company = A \wedge Cena = 2000 \wedge Cas = 48h, t_2), t_3)$

Druhý příklad je vyjednávání agentů DD a SD ohledně služby sledování zákazníkových stránek. Jak bylo zmíněno výše, agent DD má vyšší autoritu než agent SD, čili agent SD nemůže požadavek nikdy odmítnout. Agent DD požaduje, že služba musí být dokončena během 24 hodin. SD odpovídá, že jeden z jeho zaměstnanců odjíždí na dovolenou, čili služba nebude dokončena dříve než za 48 hodin. DD odpovídá tím, že vynucuje dokončení během 24 hodin. Dialog je reprezentován následující sekvencí:

1. $offer(DD, SD, Cas = 24h \wedge Sluzba = SledovaniStranky, t_1)$
2. $appeal(SD, DD, Cas = 48h, Zamestnanec(X) \wedge Dovolena(X), t_2)$
3. $appeal(DD, SD, time = 24h, time = 24h, t_3)$

V tomto okamžiku má agent SD dva protichůdné argumenty:

$$(Cas = 48h, Zamestnanec(X) \wedge Dovolena(X)), (time = 24h, time = 24h)$$

Jako podporu těchto argumentů lze uvažovat důvěryhodnost agenta, který je předložil. Vzhledem k tomu že agent DD je nadřazen agentovi SD, má jeho argument pro agenta SD větší váhu. Proto agent SD musí splnit požadavek agenta DD.

Třetí příklad uvádí komunikaci mezi agentem CSD a DD. Agent CSD požaduje navržení sítě v čase 24 hodin. Agent DD odpovídá, že síť může navrhnout v čase 48 hodin, neboť jeden z jeho zaměstnanců je právě na dovolené. Na to agent CSD reaguje výhružkou, jestliže agent DD nepřijme splnění požadované služby do 24 hodin, informuje agent CSD nadřízeného agenta DD o jeho neschopnosti.

1. $offer(CSD, DD, Cas = 24h \wedge Sluzba = Navrh, t_1)$
2. $appeal(DD, CSD, Cas = 48h, Zamestnanec(X) \wedge Dovolena(X), t_2)$
3. $threaten(CSD, DD, \neg accept(DD, CSD, Cas = 24h \wedge Sluzba = Navrh),$
   $appeal(CSD, BossOfDD, DD = nekompetentni,$
   $\neg accept(DD, CSD, Cas = 24h \wedge Sluzba = Navrh), t_1)$

# Kapitola 5

# Hierarchický model důvěry v kontextu

Tato kapitola popisuje hierarchický model důvěry v kontextu tak, jak je popsán v práci [3]. Jednotlivé sekce popisují jeho části.

V sekci 7.1 je tento model neformálně popsán. Sekce 5.2 pak tento model přesně matematicky definuje. V sekci 5.3 je pak definováno, jak vyjádřit důvěru. Sekce 5.4 se pak zabývá výpočtem důvěry v grafu, na základě vztahů mezi uzly. Poslední sekce 5.5 pak popisuje navržený model bezdrátového senzoru.

## 5.1 Neformální popis

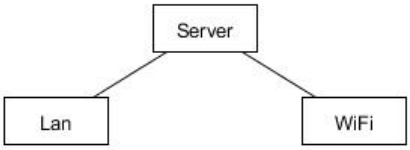
V reálném světě máme o objektech ve svém okolí jisté představy. Například víme, že dané auto je červené, nebo že konkrétní člověk je vysoký. O určitých věcech však nemusí být naše představy natolik konkrétní. Například o vodě v řece budeme předpokládat, že je studená. Tento předpoklad jsme získali na základě nějakých okolností. Například předpokládáme že je studená, neboť venku je zima. Skutečnou teplotou vody si však nemůžeme být jisti, a proto je možné vyjádřit jakousi důvěru v tento předpoklad.

Agent v systému, ve kterém je situován, také může znát několik dalších objektů. Pokud je to pro něj má význam, může si uchovávat informaci jejich stavu. Předpokládejme například, že v systému existuje server, který agent může využít pro snadnější komunikaci. To znamená, že agent je schopen provést komunikaci i bez serveru, nicméně je to pro něj nákladnější. Naproti tomu při použití serveru je komunikace levnější, nicméně prostředník v komunikaci může způsobit chyby při přenosu. Agent si tedy uchovává hodnotu důvěry, kterou v tento server má. Pokud je důvěra vysoká, agent bez problémů server využívá. Pokud je důvěra naopak nízká, agent raději provede komunikaci sám, než aby riskoval její neúspěch.

Ve výše popsaném příkladě je server jednoduchou entitou, o níž má agent představu pouze jako o celku. Jedná se o jisté zobecnění, které nemusí být ve všech případech vhodné. Server může mít možnost komunikovat s agentem více způsoby, například přes LAN a WiFi. V tomto případě je vhodné uchovávat si o serveru i informace o funkci těchto jeho podsložek. Schopnost komunikovat přes LAN, respektive schopnost komunikovat přes WiFi, udává celkovou schopnost serveru komunikovat. Nicméně pokud si agent bude uchovávat důvěru i v tyto jeho podsložky, může se rozhodnout, který způsob využije.

Dostáváme se tak k situaci, kdy objekty v okolí agenta můžeme popsat jako víceúrovňové grafy. Příklad, jak by mohl takový graf vypadat pro popsaný server, je na obrázku 5.1.

Obrázek 5.1: Víceúrovňový graf reprezentující server




Tento přístup, využívající hierarchický model důvěry v kontextu pro prvky systému byl navrhnut v článku [3]. Jeho jednotlivé části popisují následující sekce.

## 5.2 Struktura modelu

Hierarchický model důvěry v kontextu je navržen jako víceúrovňový graf. Množina N značí uzly grafu. Množina E značí hrany grafu. Přitom existuje rozdělění množiny N

$$N = N_1 \cup N_2 \cup N_3 ... \cup N_n$$

$n$ určuje počet úrovní grafu. Množiny $N_1, N_2, N_3 ... N_n$ jsou neprázdné množiny, které obsahují uzly náležící do stejné úrovně.

Množinu E, obsahující hrany, lze podobným způsobem rozdělit na neprázdné podmnožiny $E_1, E_2, E_3 ... E_{n-1}$, přičemž platí

$$E = E_1 \cup E_2 \cup E_3 ... \cup E_{n-1}$$

Hrana vždy spojuje právě dva uzly. Tyto uzly musí ležet v sousedících úrovních. Označme si hranu $e$, spojující uzly $a, b \in N$. Poté musí platit následující tvrzení

$$e \in E \Leftrightarrow ((a \in N_i) \wedge (b \in N_{i+1})) \vee ((a \in N_{i+1}) \wedge (b \in N_i))$$

Kompletní struktura pro hierarchický model důvěry v kontextu byla v práci [3] navržena následovně.

$$HMTC = (N, E, T, \rho, \omega)$$

$N$ je množina uzlů a $E$ je množina hran, pro které platí výše popsané souvislosti. $T$ je množina diskrétních časových momentů. Funkce $\rho$ přiřazuje jednotlivým uzlům v daném čase důvěru z intervalu $< 0, 1 >$. Jedná se o zobrazení

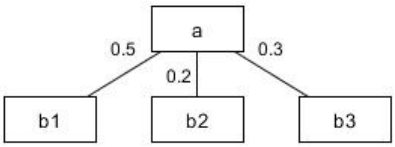
$$\rho : N \times T \rightarrow < 0, 1 >$$

Funkce $\omega$ přiřazuje každé hraně $e \in E$ váhu z intervalu $< 0, 1 >$. Jedná se tedy o zobrazení

$$\omega : E \rightarrow < 0, 1 >$$

Funkce $\omega$ je omezena pravidlem, podle kterého musí součet vah hran, spojujících uzel s uzly nižší úrovně, vždy dát součet 1. Situace je zachycena na obrázku 5.2.

Obrázek 5.2: Jednoduchý víceúrovňový graf. Uzel $a \in N_1$, uzly $b1, b2, b3 \in N_2$. Suma vah hran spojující uzel $a$ s uzly nižší úrovně je rovna 1.




## 5.3 Důvěra

V předchozím textu bylo neformálně popsáno, co to důvěra je. Jako součást struktury HMTC byla definována funkce $\rho$, která jednotlivým uzlům přiřazovala důvěru v daném čase. V této kapitole přesně definujeme, co to důvěra je, a jakým způsobem ji lze vyjádřit.

Důvěra nám vyjadřuje, jak moc v jistou věc, případně vlastnost, věříme. Běžným vyjádřením míry důvěry je hodnota z intervalu $< 0, 1 >$. Platí přitom, že čím nižší je toto číslo, tím nižší důvěru v danou věc máme. Například uveďme, že v řidičské schopnosti svého partnera máme důvěru $0, 8$. V tom případě nemáme problém této její/jeho vlastnosti využít a nechat se svézt. Pokud je však naše důvěra pouze $0, 1$, budeme pravděpodobně raději řídit sami.

Rozšířením způsobu vyjadřování důvěry pak je udávání této hodnoty jako intervalu. Místo konkrétního čísla, například $0, 5$, pak dostáváme interval, například $< 0, 4; 0, 6 >$. Tento přístup nám jistým způsobem umožňuje modelovat nejistotu nad důvěrou.

V sekci 5.2 byla definována funkce $\rho$, přiřazující jednotlivým uzlům důvěru z intervalu $< 0, 1 >$ pro konkrétní čas. Tuto funkci nyní nahradíme dvěma funkcemi. Funkce $\rho^{min}$ bude vyjadřovat dolní mez intervalu důvěry, funkce $\rho^{max}$ pak mez horní. Obě funkce definujeme jako zobrazení

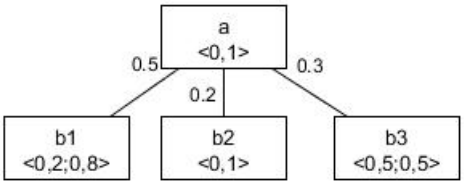
$$\rho^{min} : N \times T \rightarrow < 0, 1 >$$
$$\rho^{max} : N \times T \rightarrow < 0, 1 >$$

Zároveň musí platit podmínka

$$\forall n \in N : \rho^{min}(n) \leq \rho^{max}(n)$$

Pro upřesnění dodejme, že pokud je důvěra rovna intervalu $< 0, 1 >$, pak o příslušném uzlu nemáme žádné informace. Tuto důvěru lze považovat za výchozí stav každého uzlu v systému. Na obrázku 5.3 je zachycen systém z obrázku 5.2, tentokráte již s přiřazenou důvěrou pro každý uzel.

Obrázek 5.3: Graf z obrázku 5.2 s přiřazenou důvěrou v jednotlivých uzlech. Důvěra v uzlu $a$ je $< 0, 1 >$, v uzlu $b1 < 0, 2; 0, 8 >$, v uzlu $b2 < 0, 1 >$ a v uzlu $b3 < 0, 5; 0, 5 >$



## 5.4 Výpočet důvěry

Mějme graf hierarchického modelu důvěry v kontextu, který reprezentuje jistý systém. Jednotlivé uzly reprezentují části tohoto systému, hrany pak vztahy mezi nimi. Důvěra v každém uzlu udává, jakou máme důvěru v danou část systému. Díky hierarchickému uspořádání víme, že uzly v nižších vrstvách mají přímý vliv na uzly ve vyšší vrstvě, a naopak. Při změně hodnoty důvěry v určitém uzlu je tedy možné přepočítat hodnoty důvěry i v uzlech, které mají nějakou závislost na změněném uzlu.

Pro tento výpočet jsou v práci [3] navrženy dvě metody. Jedná se o výpočet up-direction, tedy výpočet směrem vzhůru, a down-direction, tedy výpočet směrem dolů. Přesné popisy těchto metod jsou v následujících sekcích.

### 5.4.1 Výpočet up-directon

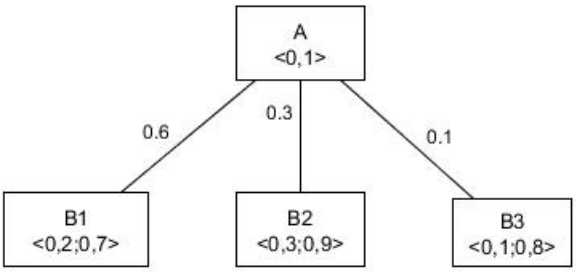
Výpočet up-direction slouží ke změně hodnoty důvěry v uzlu na základě jeho potomků. Výpočet probíhá tak, že dolní hranice intervalu důvěry rodiče je určena součtem váhovaných dolních hranic intervalu důvěry všech jeho potomků. Horní hranice je pak spočtena stejným způsobem z horních hranic intervalů potomků. Mějme uzel $A$, a uzly $B_1, B_2, ...B_n$. Rovnice pro výpočet nové důvěry v uzlu $A$ jsou následující

$$\rho^{min}(A, t^{i+1}) = \sum_{j=1}^{n} w_j \rho^{min}(B_j, t^i)$$
$$\rho^{max}(A, t^{i+1}) = \sum_{j=1}^{n} w_j \rho^{max}(B_j, t^i)$$

$w_j$ značí váhu hrany spojující uzel $A$ s uzlem $B_j$.

Průběh výpočtu si budeme ilustrovat na následujícím případě. Mějme systém, obsahující 4 uzly, a to $A$, $B1$, $B2$ a $B3$. Uzel $A$ je rodičem uzlů $B1$, $B2$ a $B3$. Hodnota důvěry v uzlu $A$ má hodnotu $< 0, 1 >$, čili je neznámá. Hodnota důvěry v uzlu $B1$ je $< 0, 2; 0, 7 >$, v uzlu $B2$ $< 0, 3; 0, 9 >$ a v uzlu $B3$ $< 0, 1; 0, 8 >$. Tuto situaci ilustruje obrázek 5.4

Obrázek 5.4: Graf systému obsahujícím uzly $A$, $B1$, $B2$ a $B3$. Hodnotu uzlu $A$ lze dopočítat výpočtem up-direction.




Novou hodnotu důvěry v uzlu $A$ tedy vypočteme dosazením do výše popsaných rovnic.

$$\rho^{min}(A, t^{i+1}) = 0,6 * 0,2 + 0,3 * 0,3 + 0,1 * 0,1 = 0,22$$
$$\rho^{max}(A, t^{i+1}) = 0,6 * 0,7 + 0,3 * 0,9 + 0,1 * 0,8 = 0,77$$

Tímto výpočtem jsme tedy dostali novou hodnotu důvěry pro uzel $A$, která má hodnotu $< 0, 22; 0, 77 >$.

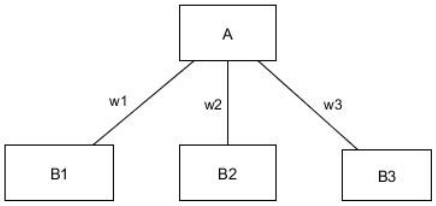
### 5.4.2 Výpočet down-direction

Výpočet down-direction slouží k aktualizaci hodnoty uzlu na základě jeho rodiče, a jeho sousedních uzlů. Mějme uzel $A$, který je rodičem uzlů $B_1, B_2, ...B_n$. Tuto situaci zachycuje obrázek 5.5. Aktualizace hodnoty $B_1$ probíhá na základě následujících rovnic

$$\rho^{min}(B_1, t^{i+1}) = \frac{\rho^{min}(A,t^i) - \sum_{j=2}^{n} w_j \rho^{max}(B_j,t^i)}{w_1}$$
$$\rho^{max}(B_1, t^{i+1}) = \frac{\rho^{max}(A,t^i) - \sum_{j=2}^{n} w_j \rho^{min}(B_j,t^i)}{w_1}$$

$w_j$ značí váhu hrany spojující uzel $A$ s uzlem $B_j$.

Obrázek 5.5: Graf systému obsahujícím uzly $A$, $B1$, $B2$ a $B3$. Hodnotu uzlu $B1$ lze určit výpočtem down-direction.



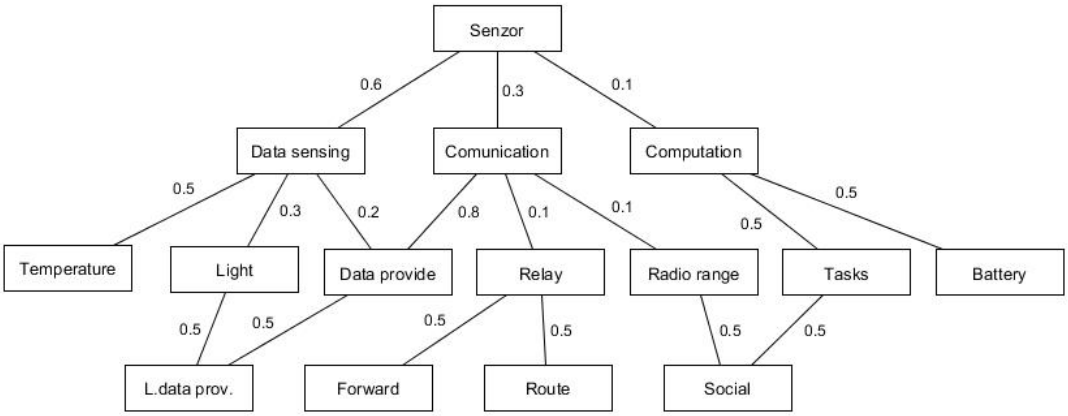

V grafu může nastat situace, kdy jeden uzel má více rodičů. V takovém případě je výpočet proveden zlvášť pro každého z nich. Tyto výsledky jsou pak agregovány průnikem. Mějme uzly $A_1$ a $A_2$, které jsou oba rodičem uzlu $B_1$. Výpočtem důvěry přes tyto rodiče dostáváme výsledky $\rho(B_1^{A_1}, t^{i+1})$ a $\rho(B_1^{A_2}, t^{i+1})$. Výslednou hodnotu v uzlu $B_1$ pak dostaneme jako

$$\rho(B_1, t^{i+1}) = \rho(B_1^{A_1}, t^{i+1}) \cap \rho(B_1^{A_2}, t^{i+1})$$

## 5.5 Navržený model

V práci [3] byl navržen hierarchický model důvěry v kontextu, který reprezentuje bezdrátový senzor. Graf má celkem 15 uzlů, přičemž každý reprezentuje jistou část senzoru. Tento model je pro tuto práci základním modelem, nad kterým probíhají veškeré simulace. Jeho graf je zachycen na obrázku 5.6.

Obrázek 5.6: Graf senzoru



# Kapitola 6

# Argumentace

Argumentace slouží k vyjednávání konfliktních situací. Jako model systému, nad kterým je argumentace prováděna, je využit hierarchický model důvěry v kontextu, popsaný v kapitole 5. V procesu argumentace jsou vždy zahrnuti dva agenti, z niž oba mají totožné představy o struktuře grafu. Vyjednávaným prvkem jsou hodnoty důvěry u jednotlivých uzlů. Pokud jsou představy agentů o této hodnotě ve stejném uzlu různé, je rozhodnuto, zda jsou konfliktní. Pokud ano, agenti se snaží argumentací dosáhnout nekonfliktního stavu.

Tato kapitola popisuje základní prvky, na kterých je vystavěn argumentační protokol. Sekce 6.1 popisuje, za jakých odolností dochází mezi uzly ke konfliktu. V sekci 6.2 jsou popsány události, jakožto základní způsob změny v grafu. Sekce 6.3 pak zavádí pojem klíčového uzlu. V sekci 6.4 definována váha důvěry pro daný uzel, a způsob výpočtu. Sekce 6.5 pak popisuje způsob, kterým jsou tvořeny argumenty pro jednotlivé uzly.

## 6.1 Konflikt

Ke konfliktu mezi jednotlivými agenty dochází v případě, že jejich představy o stejném uzlu se liší. Informace o daném uzlu je udávána hodnotou důvěry. Pokud mají agenti různé představy o důvěře v daném uzlu, může dojít ke konfliktu. Při porovnávání hodnot důvěry ve dvou uzlech může nastat několik situací.

1. Hodnoty důvěry jsou stejné (např. $< 0,2; 0,8 >$,$< 0,2; 0,8 >$). V tomto případě konflikt samozřejmě nenastává.

2. Hodnota důvěry jednoho uzlu leží v intervalu hodnoty důvěry uzlu druhého (např. $< 0,2; 0,8 >$,$< 0,3; 0,6 >$). V tomto případě opět konflikt nenastává.

3. Interval hodnoty důvěry jednoho uzlu částečně překrývá interval hodnoty důvěry druhého uzlu (např. $< 0,2; 0,6 >$,$< 0,4; 0,8 >$). Tato situace značí konflikt.

4. Intervaly hodnot důvěry uzlů mají prázdný průnik (např. $< 0,2; 0,4 >$,$< 0,5; 0,8 >$). V této situaci opět dochází ke konfliktu.

## 6.2 Události

Pokud by během života systému nedocházelo ke změnám důvěry, zůstávaly by představy agentů o jednotlivých částech neměnné. Hodnoty důvěry jednotlivých uzlů jsou nicméně

měněny na základě vnějších událostí. Agent může například po neúspěšném použití funkce snížit hodnotu důvěry uzlu, který ji reprezentuje. Dalším příkladem je změna důvěry na základě zprávy od jiného agenta. Možností, na základě kterých dochází k přehodnocení důvěry, může být hned několik.

V této práci se nicméně abstrahujeme od jejich původu. Jakákoliv vnější změna je reprezentována událostí. Událost se vždy vztahuje ke konkrétnímu uzlu a udává, na jakou hodnotu byla změněna důvěra daného uzlu, a jaká váha je události přiřazena. Událost tedy můžeme vyjádřit jako dvojici

$$event = (newTrust, weight)$$

Hodnota $newTrust$ udává novou důvěru, která je uzlu přiřazena. Hodnota $weight$ pak určuje váhu, kterou tato událost má.

Důležitou doplňkovou informací přitom je, ke kterému uzlu se daná událost vztahuje. Tato informace může být uchovávána i mimo vlastní událost, avšak v této práci je přímo součástí události. Takto rozšířenou událost vyjádříme jako trojici

$$event = (newTrust, weight, nodeID)$$

Hodnota $newTrust$ udává novou důvěru, kterou uzlu událost přiřazuje. Hodnota $weight$ určuje váhou události. $nodeID$ je pak označení uzlu, ke kterému se daná událost vztahuje.

Každý agent si uchovává historii všech událostí, které nad systémem nastaly. Na jejich základě je poté schopen argumentovat s jinými agenty.

## 6.3 Klíčové uzly

Jak již bylo řečeno dříve, události jsou způsobem, jak ovlivňovat představy agenta o systému. Zároveň se jedná o jediný možný způsob, neboť do sebe abstrahují všechny možné případy. Jakákoliv událost tak má významný vliv na celý systém.

Pro přesnou definici klíčového uzlu je nutné nejprve definovat počáteční stav systému.

**Počáteční stav systému je takový stav, kdy představy všech agentů o všech uzlech grafu jsou schodné.**

Za počáteční stav lze považovat například situaci, kdy žádný z agentů nemá informace o žádném prvku systému. Tento stav lze vyjádřit hodnotou důvěry $< 0,0; 1,0 >$ ve všech uzlech systému. Počátečním stavem ovšem může být i situace, kdy agenti mají určité představy o stavu systému, ovšem ty jsou u všech agentů schodné. Z obecnějšího hlediska můžeme říci, že počáteční stav je takový stav, kdy hodnota váhy v danou důvěru ve všech uzlech systému je rovna 0.

Jedinou možností, jak se systém může dostat z počátečního stavu, je příchod události. Událost je vztažena k danému uzlu, a jeho hodnotu důvěry změní (ve speciálním případě pouze potvrdí). Na základě nové hodnoty tohoto uzlu je přepočítána hodnota všech okolních uzlů způsobem, který je detailně popsán v sekci 5.4.

Jelikož je to právě událost nad daným uzlem, která může změnit celý systém, je uzel, pro který tato událost nastala, označen jako klíčový uzel. Klíčový uzel změní systém z jeho počátečního stavu. Klíčových uzlů může být více.

**Klíčové uzly jsou uzly takové, které mění stav systému z počátečního stavu.**

Pro argumentaci, zejména pro navržený argumentační protokol, mají klíčové uzly nezbytný význam. Hodnota důvěry v klíčovém uzlu je dokázána samotnou událostí, a jako taková je téměř nezpochybnitelná. Jedinou výjimkou může být přítomnost silnější události v některém ze sousedních uzlů.

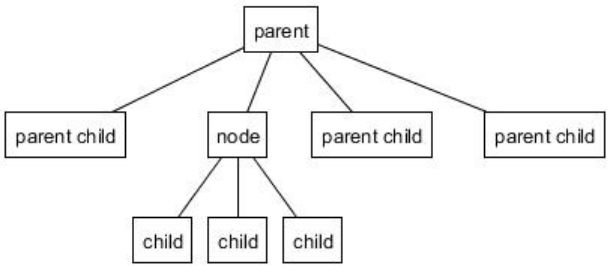
## 6.4 Váha důvěry

Každý uzel obsahuje určitou důvěru. Důvěra, kterou má daný uzel přiřazenou, musela být nějakým způsobem získána. Váha důvěry nám udává, jak moc věříme tomu, že důvěra v uzlu je právě taková, jaká je. Z jiného úhlu pohledu lze říct, že váha důvěry určuje, jak silný jsme schopni sestavit argument pro tuto důvěru v daném uzlu.

První možností je, že hodnota důvěry v uzlu byla přímo změněna událostí. V tomto případě se uzel stává uzlem klíčovým, tak jak je popsáno v sekci 6.3. Jelikož hodnoty důvěry v klíčových uzlech jsou dokazatelné váhou událostí, která pro tento uzel nastala, je váha důvěry tohoto uzlu rovna váze příslušné události.

Druhým případem je výpočet důvěry uzlu z uzlů v jeho okolí. Ten probíhá způsobem, který je popsán v sekci 5.4. Z něj je zřejmé, že každý uzel je ovlivněn svým okolím. Okolím uzlu jsou přitom myšleny všechny uzly, které jsou potomky daného uzlu, rodič tohoto uzlu, a všichni potomci rodiče. Okolí uzlu je znázorněno na obrázku 6.1.

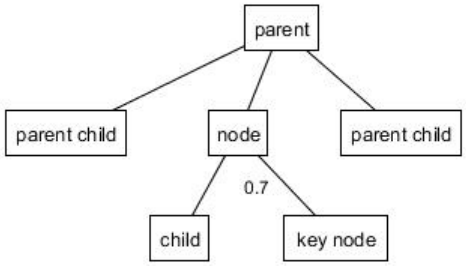
Obrázek 6.1: Příklad okolí bodu. *Node* je výchozí uzel. *Child* jsou potomci tohoto uzlu. *Parent* je rodič uzlu, a *parent child* jsou potomci rodiče.




Pokud je tedy hodnota důvěry v uzlu spočtena z jeho okolí, je z něj spočtena i jeho váha důvěry. V potaz jsou přitom brány pouze klíčové uzly, které se v tomto okolí vyskytují. Nová váha je vypočtena jako součin váhy události klíčového uzlu a hrany, kterou je počítaný uzel spojen s klíčovým. Možný výpočet je zachycen na obrázku 6.2. Uzel je spojen s klíčovým uzlem hranou s váhou 0,7. Výsledná váha důvěry v tomto uzlu bude tedy rovna

$$VahaDuvery = 0,7 * VahaUdalosti.$$

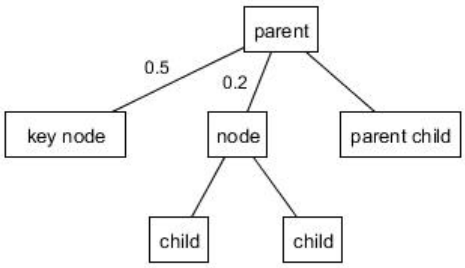
Obrázek 6.2: Příklad výpočtu váhy důvěry. Klíčový uzel je potomkem počítaného uzlu. Hrana spojující uzly má váhu 0,7.




Pokud je klíčovým uzlem potomek rodiče počítaného uzlu, je váha spočtena jako součin hrany z počítaného uzlu do jeho rodiče, z rodiče do klíčového uzlu a váhy události v klíčovém uzlu. Tato situace je zachycena na obrázku 6.3. Rodič je spojen s počítaným uzlem hranou s váhou 0,2. Klíčový uzel je spojen s rodičem hranou s váhou 0,5. Výsledná váha důvěry v tomto uzlu bude tedy rovna

$$VahaDuvery = 0,2 * 0,5 * VahaUdalosti.$$

Obrázek 6.3: Příklad výpočtu váhy důvěry. Klíčový uzel je potomkem rodiče počítaného uzlu. Hrana spojující uzel s rodičem má váhu 0,2. Hrana spojující rodiče s klíčovým uzlem má váhu 0,5.




## 6.5 Argument

Argument je základním prvkem argumentace. Agenti si předkládají argumenty jako důkazy svých tvrzení a vyvrací jimi argumenty druhých.

Argument se skládá ze závěru, čímž je v našem případě navrhovaná hodnota důvěry pro daný uzel, a z podpory argumentu. Argument je vždy konstruován k danému uzlu. Jeho závěr je hodnota důvěry v daném uzlu. Pokud se jedná o uzel klíčový, je podpora argumentu tvořena pouze událostí příslušící tomuto uzlu. Pokud uzel není klíčový, je jeho podpora vytvořena hodnotou důvěry všech uzlů v jeho okolí.

Doplňkovou informací každého argumentu je jeho váha, která určuje, s jakou váhou je argument celkově sestaven. Váha argumentu je určena váhou důvěry v uzlu, pro který je argument tvořen. Obecně můžeme tedy argument zapsat jako

$$argument = (newTrust, weight, support)$$

Hodnota $newTrust$ značí navrhovanou hodnotu důvěry pro daný uzel. $weight$ určuje sílu argumentu. Množina $support$ pak obsahuje podporu argumentu.

Na obrázku 6.4 je zachycen možný stav systému. Klíčový uzel, v obrázku označený *key node*, byl změněn událostí, která uzlu přiřazuje hodnotu $< 0,4; 0,8 >$ s váhou $0,5$.

$$udalost = (< 0,4; 0,8 >, 0,5, keynode)$$

Cílem argumentace je uzel *node*, pro který se snažíme sestavit argument. Závěrem argumentu bude hodnota důvěry $< 0,04; 0,98 >$. Jelikož uzel sám není uzlem klíčovým, bude podpora argumentu obsahovat hodnoty ze všech uzlů v okolí. Výsledná váha argumentu bude vypočtena tak, jak je popsáno v předešlé sekci. Výsledkem tedy bude váha

$$vaha = 0,1 * 0,5 = 0,05$$

Jak již bylo řečeno, podpora argumentu bude obsahovat hodnoty všech uzlů v okolí, neboť diskutovaný uzel není klíčový. Pro jednoduchost lze jednotlivé položky podpory zasílat ve stejném formátu, jaký mají události, tedy

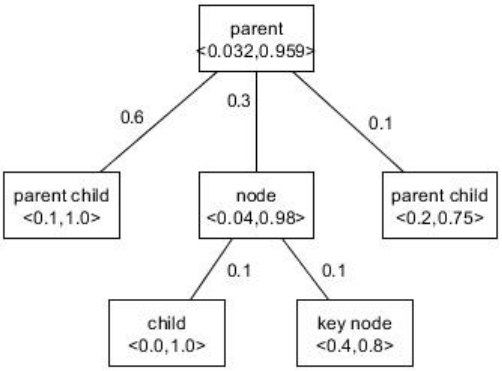
$$polozkaPodpory = (newTrust, weight, nodeID)$$

Hodnota $newTrust$ udává důvěru, kterou daný uzel obsahuje. Hodnota $weight$ určuje váhu důvěry. $nodeID$ je pak označení uzlu.

Pro situaci na obrázku 6.4 je tedy podpora vytvářeného argumentu následující

$$support = \{(< 0,0; 1,0 >; 0,005; child), (< 0,4; 0,8 >; 0,5; keynode),$$
$$(< 0,032; 0,959 >, 0, parent), (< 0,0; 1,0 >, 0, parentchild1),$$
$$(< 0,2; 0,75 >, 0, parentchild2)\}$$

Obrázek 6.4: Příklad možného stavu systému v okamžiku tvorby argumentu.



# Kapitola 7

# Argumentační protokol

Argumentační protokol slouží jako prostředek k argumentaci. Díky němu jsou agenti schopni si navzájem porozumět, a reagovat tak na zprávy od druhého agenta. Návrh tohoto protokolu je podstatnou částí této práce. Základní pojmy pro pochopení jeho činosti jsou popsány v kapitole 6.

Argumentační protokol je navržen pouze za účelem argumentace nad reputačním systémem, který je popsán v kapitole 5. S jeho obecným nasazením pro jiné problémy nebylo počítáno, a proto k těmto účelům není vhodný.

Tato kapitola popisuje navržený argumentační protokol. V sekci 7.1 neformálně popsána funkčnost protokolu. Následující sekce pak podrobně rozebírají jednotlivé zprávy protokolu. V sekci 7.2 je popsána zpráva assert, v sekci 7.4 zpráva accept. V sekci 7.3 je pak popsána zpráva challenge a v sekci 7.5 zprávy end, a endconfirm. V poslední sekci 8.2 je pak uveden praktický příklad komunikace.

## 7.1 Popis protokolu

Základem protokolu jsou zprávy, které si agenti mezi sebou zasílají. Je využito celkem 5 zpráv, kterými jsou

- Assert
- Accept
- Challenge
- End, Endconfirm

Argumentace vždy probíhá nad konkrétním uzlem. Během procesu je však procházeno i okolí uzlu. Pokud jsou v okolí nalezeny další uzly v konfliktu, je o hodnotě jejich důvěry také argumentováno. Tento přístup slouží k vyjednání hodnoty důvěry v uzlu s co nejvýšší váhou.

Argumentace začíná zasláním zprávy *Assert* od jednoho agenta druhému agentovi. Touto zprávou agent předkládá svoji představu o konkrétním uzlu. Po zaslání této zprávy agent čeká na odpověď od druhého agenta. Ten může zaslat jednu ze tří zpráv, a to *Accept*, *Assert* nebo *Challenge*.

Agent odpoví na zprávu *Assert* zprávou *Accept* v případě, že hodnoty důvěry v uzlu nejsou v konfliktu. Zprávu *Accept* může zaslat také v případě, kdy uzly v konfliktu jsou, nicméně agent není schopen napadnout žádnou položku z podpory argumentu.

Zpráva *Assert* může být odpovědí na přijatou zprávu *Assert* v případě, kdy agent nalezne v podpoře zaslaného argumentu uzel, jehož hodnotu důvěry může zpochybnit. Tento případ nastane v momentě, kdy agent má pro daný uzel vyšší váhu důvěry než je ta, kterou druhý agent zaslal v podpoře argumentu. Druhým případem, kdy je zpráva *Assert* odpovědí na zprávu *Assert* je situace, kdy uzly jsou v konfliktu, a přijímající agent dokáže pro svou hodnotu důvěry sestavit silnější argument.

Odpověd *Challenge* je zaslána po obdržení zprávy *Assert* v případě, že agent v podpoře argumentu nalezne uzel, jehož váha důvěry je vyšší než jakou agent sám má.

Z tohoto popisu vyplývá, že v případě, kdy jsou uzly v konfliktu, je argumentováno nad celou podporou původního argumentu. Pokud agent, který dostal původní zprávu *Assert*, zná vyšší váhu důvěry pro určitý uzel z podpory argumentu, zašle ji jako návrh druhému agentovi zprávou *Assert*. Pokud naopak pro daný uzel má nižší váhu důvěry, požádá druhého agenta o předložení argumentu o tomto uzlu zprávou *Challenge*. Z tohoto důvodu je nutné, aby si každý agent pamatoval uzly, které byly součástí podpor dříve zaslaných argumentů. Ty pak tvoří vyjednávací pamět.

Zprávami *Accept* a *End* není zasílán žádný argument. Po obdržení těchto zpráv zkontroluje agent svou vyjednávací pamět. Pokud ta obsahuje nějaké položky, jsou zaslány příslušné zprávy *Assert*, respektive *Challenge*.

Zpráva *Challenge* slouží k vyžádání konstrukce argumentu k danému uzlu. Zpráva *Endconfirm* pak slouží k ukončení komunikace.

Následující sekce obsahují detailní popis činosti jednotlivých zpráv.

## 7.2 Assert

Zpráva *Assert* slouží k předložení argumentu o určitém uzlu. Agent zasílá svou představu o tom, jakou hodnotu má důvěra v určitém uzlu, což podkládá vytvořeným argumentem. Agent který tuto zprávu přijme nejprve určí, zda navrhovaná hodnota důvěry je v konfliktu s důvěrou, kterou agent v uzlu sám má. Pokud nenastává konflikt, odpoví agent zprávou *Accept*, a nedochází k žádné změně ani u jednoho z agentů.

Pokud jsou hodnoty důvěry v konfliktu, je porovnána váha argumentu s váhou důvěry, kterou přijímající agent v uzlu má. Pokud má argument větší váhu, je důvěra navrhovaná zasílajícím agentem akceptována.

Jestliže byla navrhovaná hodnota akceptována, je procházena podpora argumentu. Přijímající agent hledá, zda pro některý z uzlů obsažených v argumentu nemá důvěru z větší váhou. Může se zdát nelogické, že poté, co je akceptována hodnota navrhovaná argumentem, je teprve procházena podpora argumentu. Tento přístup je nicméně zvolen pro to, že i v případě, kdy přijímající agent může napadnout část podpory argumentu, nevyvrátí tím argument jako celek. Pokud tedy agent najde v podpoře uzel, který dokáže napadnout, vytvoří argument pro tento uzel, a zašle druhému agentovi zprávu *Assert*. Tímto způsobem je umožněno komplexní vyjednávání se snahou dosáhnout nejlepšího řešení.

Pokud navrhovaná hodnota nebyla na základě váhy akceptována, zašle agent zprávu *Assert* s argumentem, který pro daný uzel dokáže sám sestavit.

## 7.3 Challenge

Jestliže agent obdrží zprávu *Assert* a prochází si podporu argumentu, může nalézt uzel, jehož váha důvěry je vyšší než ta, kterou agent sám zná. V tomto případě se agent zeptá

na argument, který tuto váhu podporuje. Uloží si proto neprojítý zbytek podpory argumentu zaslaného správou *Assert* do vyjednávací paměti a zašle druhému agentovi zprávu *Challenge*, kterou ho vyzívá k předložení argumentu o konkretním uzlu.

Při obdržení zprávy *Challenge* agent zkonstruuje argument pro daný uzel, a zašle jej zprávou *Assert* druhému agentovi.

## 7.4 Accept

Zasláním zprávy *Accept* agent oznamuje druhému agentovi, že souhlasí s jeho návrhem.

Přijímající agent zkontroluje, zda jeho vyjednávací pamět obsahuje nějaké prvky. Pokud ano, porovná první z nich se svými znalostmi a zašle příslušnou zprávu *Assert*, respektive *Challenge* druhému agentovi.

Pokud již nemá žádné uzly k argumentaci, navrhne konec komunikace zasláním zprávy *End*.

## 7.5 End, Endconfirm

Tyto zprávy slouží k ukončení komunikace. Zasláním zprávy *End* agent oznamuje, že již nemá žádné další uzly, o kterých chce diskutovat.

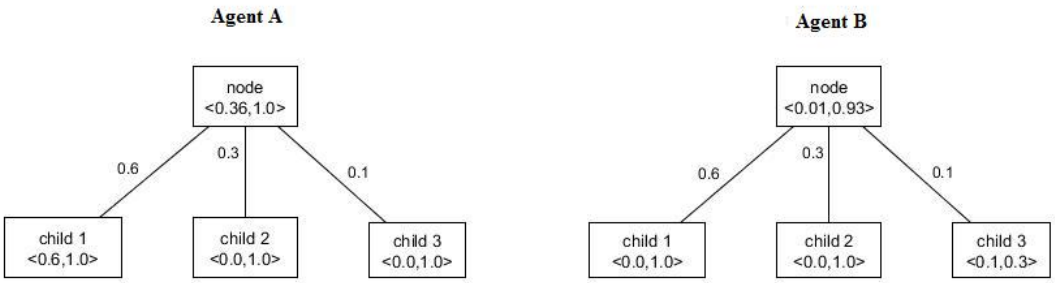
Po přijmutí zprávy *End* agent zkontroluje, zda jeho vyjednávací pamět obsahuje nějaké prvky. Pokud ano, porovná první z nich se svými znalostmi a zašle příslušnou zprávu *Assert*, respektive *Challenge* druhému agentovi.

Pokud již nemá žádné uzly k argumentaci, potvrdí konec komunikace zasláním zprávy *Endconfirm*.

## 7.6 Příklad komunikace

Možný průběh komunikace při vyjednávání budeme ilustrovat na následujícím případě. Máme dva agenty. Oba agenti znají stejný systém, který obsahuje čtyři uzly. Graf tohoto systému je znázorněn na obrázku 7.1.

Obrázek 7.1: Představy agentů o příkladovém systému




Agent A zná jednu událost, která nad systémem nastala. Tato událost změnila důvěru uzlu *child 1* na hodnotu $< 0,6; 1,0 >$ s váhou $0,5$. Přesně ji můžeme zapsat následovně:

$$udalostAgent1 = (< 0,6; 1,0 >; 0,5; child1)$$

Pro agenta A je tedy uzel *child 1* klíčovým uzlem, ze kterého je spočtena hodnota uzlu *node*. Váha důvěry v tomto uzlu je pak spočtena jako součin váhy hrany a váhy události.

$$vahaDuvery = 0,5 * 0,6 = 0,3$$

Agent B zná také jednu událost. Tato událost změnila důvěru uzlu *child 3* na hodnotu $< 0,1; 0,3 >$ s váhou $0,5$. Událost zapíšeme následovně:

$$udalostAgent2 = (< 0,1; 0,3 >; 0,5; child3)$$

Pro agenta B je tedy klíčovým uzlem uzel *child 3*. Z něj je přepočtena hodnota uzlu *node*. Váha důvěry v uzlu *node* je pak spočtena stejným způsobem jako u prvního agenta, tedy

$$vahaDuvery = 0,5 * 0,1 = 0,05$$

Argumentace agentů bude probíhat ohledně uzlu *node*, jehož hodnota je mezi agenty v konfliktu. Argumentaci zahájí jeden z agentů. Pro její výsledek není podstatné, který z nich začne. V tomto příkladu uvažujme, že argumentaci zahájí agent B zasláním své představy o tom, jak vypadá důvěra v tomto uzlu.

Agent B zasílá agentovi A zprávu *Assert*, která obsahuje argument pro navrhovanou hodnotu důvěry. Konstrukce argumentu je detailně popsána v sekci 6.5. V rámci tohoto příkladu je nicméně důležitá síla argumentu, která je rovna váze důvěry v uzlu *node*. Její výpočet je zapsán výše, hodnota váhy je $0,05$.

Agent B zasílá agentovi A: *Assert*

Agent A přijme zprávu *Assert*. Zjistí, že navrhovaná hodnota je v konfliktu s hodnotou, kterou sám zná. Jeho známá hodnota důvěry však může být podpořena argumentem se váhou $0,3$, jak je ukázáno v předešlém výpočtu. Agent A proto odpoví agentovi B zprávou ASSERT, ke které přiloží svůj argument podporující jeho hodnotu důvěry s vyšší vahou.

Agent A zasílá agentovi B: *Assert*

Agent B přijme zprávu *Assert*. Hodnoty důvěry jsou v konfliktu, a jeho váha důvěry je pro daný uzel nižší. Přijme tedy závěr argumentu, a následně prochází podporu argumentu. Prvním uzlem v podpoře je uzel *child 1*. Váha důvěry, kterou zná Agent B pro tento uzel je nižší než ta, která je obsažená v podpoře argumentu. Agent B zjistí, že uzly nejsou v konfliktu, a proto je dále neřeší.

Stejná situace jako u uzlu *child 1* nastává u uzlu *child 2*. Agent B z podpory argumentu zjistí, že váha důvěry je zde vyšší než sám zná. Uzly ovšem nejsou v konfliktu, a proto není o jejich důvěře dále diskutováno.

Pro uzel *child 3* Agent B zjistí, že zná důvěru s vyšší váhou než je ta, která je v argumentu navržena. Zašle proto agentovi A zprávu *Assert* s argumentem, podporujícím hodnotu důvěry v tomto uzlu.

Agent B zasílá agentovi A: *Assert*

Agent A přijme zprávu *Assert*. Jelikož však hodnoty důvěry nejsou v konfliktu, odpoví agentovi B zprávou *Accept*.

Agent A zasílá agentovi B: *Accept*

Agent B již nemá další uzly o kterých by se dalo diskutovat, navrhne tudíž konec komunikace, který Agent A potvrdí.

Agent B zasílá agentovi A: *End*
Agent A zasílá agentovi B: *Endconfirm*

Výsledkem argumentace je tedy dohoda mezi agenty na hodnotě důvěry v uzlu *node*, která je $< 0,36; 1,0 >$. Během této argumentace nebylo nutno zahájit argumentaci o některých uzlech z podpor argumentů, neboť žádné z nich nebyly v konfliktu.

Jako jistou nevýhodu lze chápat, že agenti si nebyly schopni vyměnit informace o klíčových uzlech právě proto, že jejich hodnoty nebyly v konfliktu s uzly druhého agenta. Nicméně pro upřesnění dodejme, že pokud by si agenti hodnoty klíčových uzlů mezi sebou vyměnili, byla by výsledná hodnota důvěry v uzlu *node* $< 0,37; 0,93 >$, což je hodnota, která není v konfliktu s dosaženým výsledkem.

# Kapitola 8

# Dosažené výsledky

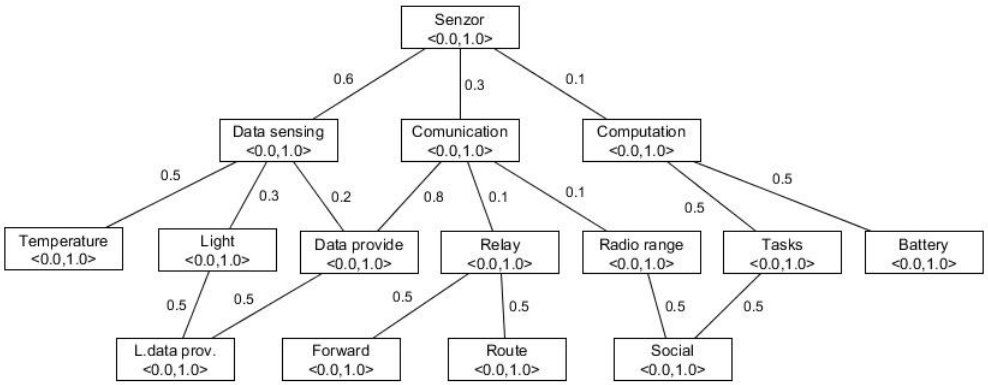
Pro testování navrženého protokolu byl vytvořen program. Základním systémem nad kterým testování probíhalo, je systém senzoru, který je stručne popsán v sekci 5.5.

Tato kapitola popisuje výsledky, kterých bylo pomocí protokolu dosaženo. V sekci 8.1 je popsán modelový systém, nad kterým probíhaly experimenty. Sekce 8.2 pak detailně popisuje příklad argumentace mezi dvěma agenty.

## 8.1 Modelový systém

Základním modelem pro testování funkčnosti argumentačního protokolu je model systému senzoru. Struktura tohoto systému je zachycena na obrázku 8.1. Jednotlivé uzly reprezentují části, respektive funkce senzoru. Uzly jsou hiearchicky uspořádány, přičemž uzly na nižší vrstvě mají vliv na ty uzly vyšší vrstvy, se kterými jsou spojeny hranou. Důležitost tohoto vlivu je pak udána váhou této hrany. Systém na obrázku 8.1 je v počátečním stavu, kdy všechny uzly mají hodnotu důvěry $< 0,0; 1,0 >$. Tento stav lze neformálně označit za situaci, kdy nemáme o systému žádné informace.

Obrázek 8.1: Systém senzoru




Hodnoty důvěry v jednotlivých uzlech lze změnit dvěma způsoby. Jedním z nich je událost. Její struktura a funkčnost jsou popsány v sekci 6.2. Jen tedy zopakujme, že událost je abstrakcí pro vnější změnu hodnoty uzlu.

Druhým způsobem je pak přepočet hodnoty důvěry podle hodnot v okolních uzlech. Přesný průběh tohoto výpočtu je popsán v sekci 5.4.

Výsledky, kterých navržený způsob argumentace nad tímto systémem dosahuje, budeme prezentovat v následujících sekcích.

## 8.2 Příklad

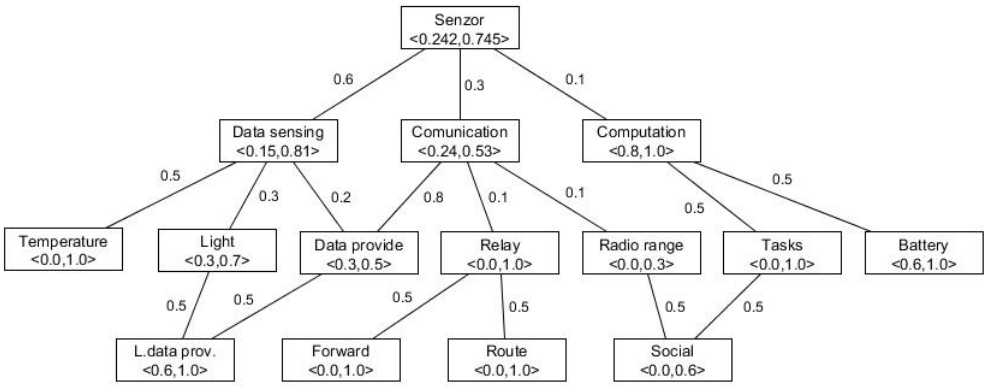
Mějme dva agenty, kteří oba znají stejný systém. Tím je právě systém senzoru, zobrazený na obrázku 8.1. Každý agent má však jiné představy o stavu systému. Učelem argumentace je vyřešit všechny možné konflikty.

Agent A zná systém ve stavu, který je zachycen na obrázku 8.2. Tento systém je ovlivněn třemi událostmi, které jsou následující

$$udalost1 = (< 0,3; 0,7 >; 0,6; Light)$$
$$udalost2 = (< 0,8; 1,0 >; 0,4; Computation)$$
$$udalost3 = (< 0,0; 0,3 >; 0,9; RadioRange)$$

Hodnoty důvěry ve veškerých ostatních uzlech jsou pak dopočítány z těchto událostí.

Obrázek 8.2: Představa agenta A o systému




Představy agenta B o systému jsou znázorněny na obrázku 8.3. Systém agenta B je ovlivněn dvěma událostmi, kterými jsou

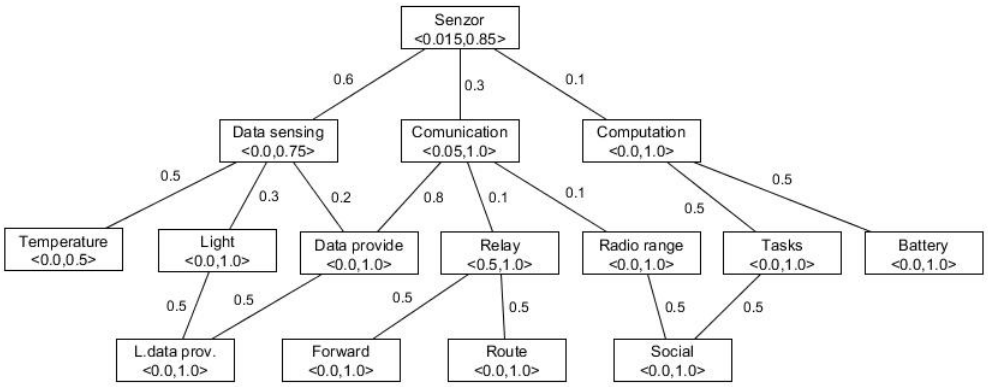
$$udalost1 = (< 0,0; 0,5 >; 0,5; Temperature)$$
$$udalost2 = (< 0,5; 1,0 >; 0,7; Relay)$$

Hodnoty důvěry ve všech ostatních uzlech jsou stejně jako u agenta A určeny výpočtem.

Jak je možné z obázků vidět, mezi představami agentů se vyskytuje pouze jeden uzel, který je v konfliktu. Tímto uzlem je *Data sensing*, jehož důvěru agent A vyhodnotil jako $< 0,15; 0,81 >$ a agent B jako $< 0,0; 0,75 >$. Argumentace tedy bude probíhat nad tímto uzlem.

Komunikaci zahájí agent B zasláním zprávy *Assert*, kterou navrhuje hodnotu důvěry v uzlu *Data sensing* $< 0,0; 0,75 >$. Vytvořený argument má váhu 0,25, zatímco agent A

Obrázek 8.3: Představa agenta B o systému




dokáže svou hodnotu potvrdit argumentem pouze s vahou 0,204. Navrhovanou hodnotu tedy akceptuje a následně prochází podporu argumentu.

V podpoře argumentu se dostane k uzlu *Light*. Pro agenta A se jedná o klíčový uzel. Jeho hodnota nicméně není v konfliktu s hodnotou, kterou zná agent B. Agent B proto tuto hodnotu nepřijme.

V průběhu diskuze je dalším zajímavým uzlem *Senzor*. Hodnoty tohoto uzlu nejsou mezi jednotlivými agentmi v konflitku. Agent B nicméně svou hodnotu v tomto uzlu může podložit pouze argumentem s váhou 0. V této situaci tedy přijme hodnotu navrhovanou agentem A, pokud ji ten může podložit argumentem s váhou větší než 0.

Diskuzí o uzlu *Senzor* je následně také otevřena diskuze o uzlu *Computation*, neboť ten je součástí podpory argumentu o uzlu *Senzor*. Agent B zná v tomto hodnotou pouze důvěru s váhou 0. Přijme tedy návrh agenta A ohledně tohoto uzlu.

Jelikož uzel *Computation* je v případě agenta A uzlem klíčovým, a jako takový je jeho argument tvořen pouze událostí, která jej ovlivnila, je argumentace ukončena. Uzel *Battery* již není diskutován, i přesto, že v případě kdy by diskutován byl, agent B by přijal hodnotu navrhovanou agentem A. Tato argumentace však již není podstatná, neboť agent B může hodnotu uzlu *Battery* snadno dopočítat.

Následující sekce diskutují představy agentu o systému po proběhnuté argumentaci.

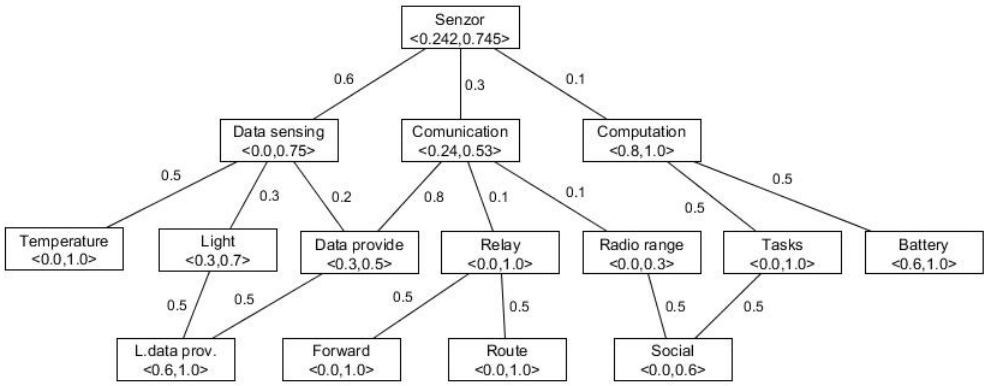
### 8.2.1 Představy agenta A

V předchozím textu byl diskutován průběh argumentace mezi agenty. Nyní se podíváme na to, jakých výsledků bylo touto argumentací dosaženo. Představy agenta A o systému po ukončení argumentace jsou na obrázku 8.4.

Porovnejme jej se stavem, ve kterém do argumentace vstupoval. Jediným uzlem, který změnil svou hodnotu je právě diskutovaný uzel *Data sensing*. Jeho hodnota ovšem v tomto systému není zcela správná, neboť je evidentně neovlivněna hodnotou uzlu *Light*. Tato situace nastala z toho důvodu, že agent A pouze přijal návrh agenta B o tomto uzlu.

Řešením je aktualizace hodnoty uzlu *Data sensing* na základě hodnoty uzlu *Light*. Tímto výpočtem dostáváme důvěru v uzlu *Data sensing* $< 0,15; 0,75 >$. Dalším výpočtem je pak upravena hodnota uzlu *Senzor* na důvěru $< 0,242; 0,709 >$. Ani jedna z těchto vypočítaných hodnot není v konfliktu s hodnotou, kterou měl systém po ukončení argumentace.

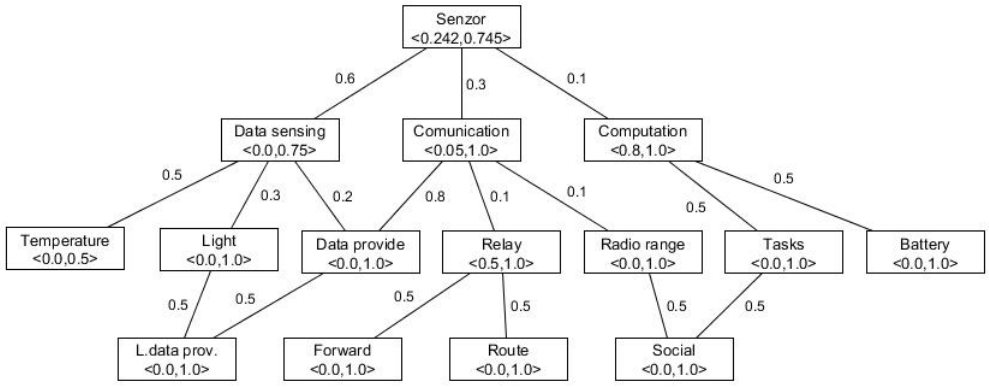
Obrázek 8.4: Představy agenta A o systému po ukončení argumentace




### 8.2.2 Představy agenta B

Zajímavější situace než u agenta A nastává u agenta B. Průběhem komunikace totiž změnil hodnotu dvou svých uzlů, a to *Senzor* a *Computation*. Představy agenta B o systému po ukončení argumentace jsou na obrázku 8.5

Obrázek 8.5: Představy agenta B o systému po ukončení argumentace

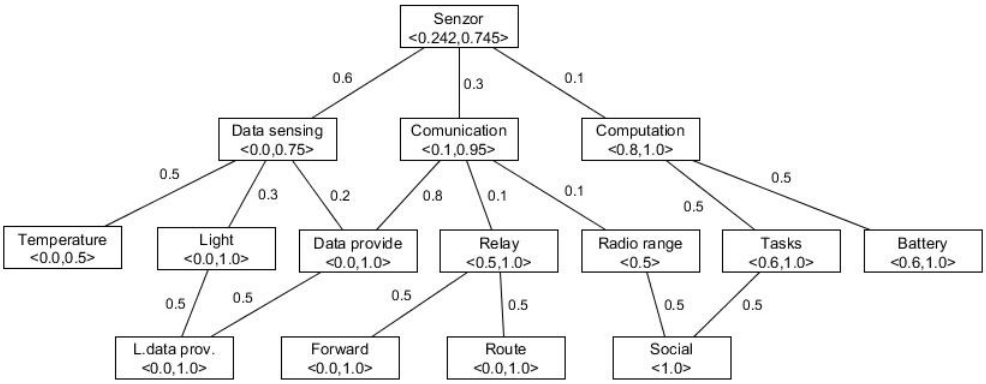



Z obrázku 8.5, že argumentací nově získaná hodnota uzlu *Computation* významně ovlivňuje okolní uzly. Jejich hodnoty je tedy nutné dopočítat. Při výpočtu a simulaci pak zjistíme, že mezi ovlivněné uzly budou patřit *Tasks*, *Battery*, *Social*, *Radio range* a *Comunication*. Výsledný systém po ukončení tohoto výpočtu je zachycen na obrázku 8.6

### 8.2.3 Stav po argumentaci

Nyní budeme diskutovat stav, který nastal po ukončení argumentace a dopočítání hodnot některých uzlů tak, jak bylo popsáno v předešlích sekcích. Porovnejme tedy představy jednotlivých agentů. Zjistíme, že se zde opět vyskytuje konflikt. Agent A má pro uzel *Radio range* důvěru $< 0,0; 0,3 >$, a tento uzel je pro něj uzlem klíčovým. Naproti tomu agent B dopočítal důvěru tohoto uzlu jako $< 0,5 >$. Stejně tak nastává konflikt v uzlu *Social*, kde agent A má hodnotu důvěry $< 0,0; 0,6 >$, zatímco agent B $< 1,0 >$.

Obrázek 8.6: Představy agenta B o systému po ukončení argumentace a aktualizace hodnot okolních uzlů na základě nově nabyté hodnoty uzlu *Computation*




Vznikem nových uzlů, které jsou v konfliktu, jsme se tedy dostali do zajímavé situace. Argumentace sice dokázala vyřešit konflikt v původně argumentovaném uzlu, nicméně v jejím průběhu vznikli nové uzly, které jsou v konfliktu. Dá se tedy říci, že navržený argumentační protokol splnil svůj účel pouze částečně. To je také jednou z jeho nevýhod. Tento problém lze ovšem vyřešit novou argumentací nad uzly v konfliktu.

Novou argumentaci spustíme nad jedním z uzlů, které jsou v konfliktu, tedy buď *Radio range* nebo *Social*. Zde se dostáváme opět do problematické situace. Jestliže spustíme argumentaci nad uzlem *Radio range*, převezme agent B hodnotu důvěry, kterou má v uzlu agent A. Jelikož se však pro agenta A jedná o uzel klíčový, bude jeho podpora tvořena pouze příslušnou událostí. Tím nedojde k vyjednávání o okolních uzlech. Hodnota uzlu *Social* tak zůstane konfliktní.

Pokud však novou argumentaci spustíme nad uzlem *Social*, bude podpora argumentu o tomto uzlu obsahovat i uzel *Radio range*. V tomto případě si agenti vymění správně obě hodnoty. Dostáváme se tak do situace, kdy v závislosti na tom, o kterém uzlu budeme argumentovat, dostáváme různé výsledky. Tato vlasnost navrženého protokolu je jednou z jeho nevýhod.

# Kapitola 9

# Možnosti rozšíření

Tato kapitola diskutuje možnosti rozšíření navrženého argumentačního protokolu. Jelikož tato práce vychází pouze z teoretických principů argumentace, jejíž implementaci v praxi takřka nenalezneme, bylo nutné takřka všechny součásti protokolu navrhnout. Takovýto komplexní návrh bez dalších podkladů sebou samozřejmě přináší spoustu rizik, která se pak projevý v nedostatcích protokolu.

Následující sekce obsahují popis jednotlivých problémů protokolu. V sekci 9.1 jsou diskutovány nevýhody klíčových uzlů a jejich možné odstranění. Sekce 9.2 popisuje nedostatky ve způsobu konstrukce argumentů. V sekci 9.3 je diskutována nepopiratelnost událostí. Sekce 9.4 pak popisuje vznik nových konfliktů v průběhu argumentace.

## 9.1 Klíčové uzly

Klíčové uzly jsou jedním se základních nedostatků navrženého protokolu. Ačkoliv jejich význam je zřejmý, přináší sebou i několik problému.

Jedním ze zásadních problémů klíčových uzlů může být jejich vysoký počet. Uvažujme extrémní situaci, kdy všichni agenti mají všechny uzly jako klíčové. Jakákoliv argumentace pak bude probíhat pouze výměnou informací o jednom uzlu, přičemž vždy zvítězí argument podložený událostí s větší váhou. Tato situace pak omezuje jakékoliv další vztahy mezi uzly.

Další nevýhodou je neschopnost algoritmu převést klíčový uzel na neklíčový. Klíčové uzly jsou určeny událostmi. Pokud by agent v jistém případě potřeboval, aby se z klíčového uzlu stal uzel neklíčový, nemá k této operaci prostředky. Tato vlastnost má také vliv na to, že uzly, jejiž hodnota je určena výsledkem argumentace, nejsou nastavovány jako klíčové. Pokud by tak bylo učiněno, mohla by složitější argumentace skončit ve stavu, kdy by všechny uzly byly klíčové.

Z výše popsaných vlastností vyplývá, že navržený protokol využívající klíčových uzlů není příliš v hodný pro dlouhodobě běžící systémy. Pokud by mezi agenty docházelo během jejich života k desítkám až stovkám argumentací, byl by navržený protokol pro takovýto systém nepoužitelný.

### 9.1.1 Dynamické klíčové uzly

Možností, jak odstranit výše uvedené problémy klíčových uzlů, by bylo zavedení dynamických klíčových uzlů. V tomto případě by protokol, případně agenti, měl prostředky ke změně klíčového uzlu na neklíčový a naopak.

Důležitou otázkou by přitom bylo, které uzly jako klíčové zvolit, případně které jako klíčové zrušit. Počet klíčových uzlů by přitom zřejmě nesměl přesáhnout jistou hranici, po které by již významně omezoval argumentaci. Možným řešením tohoto problému by mohla být dynamická změna počátečního stavu.

Co je počáteční stav bylo popsáno v sekci 6.2. Na jeho základě byly definovány klíčové uzly. Pro dlouhodobější funkčnost protokolu by bylo možné počáteční stav dynamicky měnit za tím účelem, abychom mohli redukovat klíčové uzly. Na začátku života systému by byl jistý, definovaný počáteční stav. Z něj by se stav změnil na jiný příchodem události. Agenti by o novém stavu argumentovali, přičemž by dosáhly nového, nekonfliktního stavu. Tento stav by byl poté označen za nový počáteční, přičemž veškeré události by byly smazány a všechny klíčové uzly by byly označeny jako neklíčové. Takovýto stav by se následně mohl změnit dalšími událostmi, načež by následovala další argumentace mezi agenty.

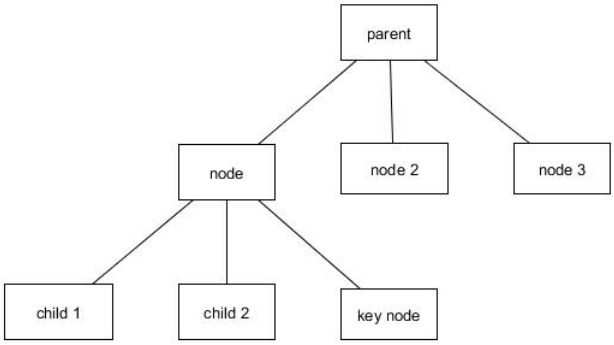
Tento přístup je však navržen čistě teoreticky, a možnost jeho funkčnosti, jeho výhody a nevýhody by vyžadovaly další výzkum.

## 9.2 Konstrukce argumentů

Argumenty jsou základním prvkem argumentačního protokolu. Podrobný popis konstrukce argumentu je popsán v sekci 6.5. K takto navrženému způsobu je vázána i nevýhoda.

Nedostatkem návrhu konstrukce argumentů je malé okolí, na které má vliv klíčový uzel. Pokud sestrojujeme argument pro uzel, který leží dále než sahá okolí příslušného klíčového uzlu, má již tento argument váhu 0. Tato situace ovšem není ideální, neboť hodnoty okolních uzlů můžeme doložit argumenty s vyšší váhou, přičemž hodnotu příslušného uzlu jsme pravděpodobně odvodili z jejich hodnot. Tuto situaci ilustruje obrázek 9.1.

Obrázek 9.1: Příklad nevýhod konstrukce argumentu k uzlu *parent*. Jeho hodnota je zcela jistě odvozena od uzlu *node*, nicméně váha argumentu je přesto 0.




Z obrázku 9.1 je vidět, že při konstrukci argumentu pro uzel *parent* se dostáváme do nevýhodné situace. Součástí podpory tohoto argumentu je uzel *node*, jehož hodnota může být podložena určitou vahou, která je závislá na váze události vztahující se ke klíčovému uzlu *key node*. Uzel *parent* byl při tom zajisté ovlivněn hodnotou uzlu *node*, a proto nelze říct, že o jeho hodnotě nemáme žádnou představu. V zkonstruovaném argumentu se nicméně toto neprojeví, neboť bude zkonstruován s váhou 0.

Tento problém lze odstranit lepším způsobem výpočtu váhy argumentů v těchto situacích. Problémem, který by však zřejmě vyvstal, je cyklické zlepšování váhy argumentů pro jednotlivé uzly. Mějme uzly *node* a *parent* z obrázku 9.1. Uvažujme možnost, že argument

sestrojený k uzlu *parent* má váhu různou od 0, která je závislá na váze důvěry uzlu *node*. Hodnotu důvěry v uzlu *node* je pak možné zvýšit, neboť je podložena nejen uzlem *key node*, ale také uzlem *parent*. Ačkoliv v případě takto jednoduchého systému se nemusí jednat o neřešitelný problém, pro systémy se složitějšími vztahy nemusí být řešení triviální.

## 9.3 Popiratelnost událostí

Na počátku je systém v počátečním stavu. Ten je změněn příchodem události tak, jak je popsáno v sekci 6.2. Událost má vždy určitou váhu, se kterou nastává. V reálném světě je samozřejmě možné, že i událost s velmi vysokou váhou nemusí být správná. Bylo by tedy vhodné, aby protokol měl prostředky k rušení událostí.

Navržený protokol tuto vlastnost nemá. Události jsou brány jako nepopiratelné, z čehož vychází i zavedení klíčových uzlů v závislosti na událostech. Ideálním řešením by bylo rozšíření protokolu o argumentaci přímo nad událostmi. Agenti by si vyměňovali informace o událostech, přičemž by se je pokoušeli podložit, případně vyvrátit argumenty. Výsledný stav systému by pak byl určen množinou událostí, na které se agenti shodli.

## 9.4 Odstranění vzniku nových konfliktů

Argumentace vždy probíhá nad určitým uzlem. Pokud jsou jeho hodnoty v konfliktu, dochází k výměně argumentů. Tato výměna však ovlivňuje i další uzly, jejíž hodnota může být změněna. U složitějších systému se tak může stát, že vyjednáním nekonfliktního stavu u jednoho z uzlů zavedeme konfliktní stav u uzlu jiného. Tato situace nastala u příkladu v kapitole 8.

Samozřejmě by bylo vhodné, aby po dokončení argumentace byly všechny uzly u obou systému v nekonfliktním stavu. Toho by mohlo být dosaženo například zvláštní kontrolou před ukončením argumentace, kdy by se prošli všechny uzly, které se v argumentech vyskytly. Pokud by mezi nimi byly nalezeny uzly v konfliktu, byla by nad nimi zahájena nová argumentace.

# Kapitola 10

# Závěr

Tato práce se zabývala vyjednáváním a argumentací mezi agenty. Základem práce byl hierarchický model důvěry v kontextu, který reprezentoval znalosti agentů o jistém systému. Mezi nimi pak měla probíhat argumentace, řešící možné konflikty.

Byly navrženy a detailně popsány dílčí prvky, které jsou součástí argumentace. Následně byl popsán argumentační protokol. Jeho funkčnost byla předvedena na praktickém příkladu, využívající hierarchický model důvěry v kontextu senzoru. Následně byly diskutovány některé z nedostatků protokolu.

Navrhnutý argumentační protokol je založen na klíčových uzlech, což jej do jisté míry omezuje. Jejich existence je v rámci protokolu popsaného touto prací nezbytná, nicméně pro dlouhodobější funkčnost protokolu nejsou vhodné. Způsob, kterým by bylo možné upravit práci s klíčovými uzly, byl také teoreticky navržen. Jeho aplikace a funkčnost jsou nicméně vhodným předmětem dalšího studia.

Na prezentovaném příkladu byla vidět funkčnost navrženého protokolu. Ačkoliv bylo argumentací dosaženo nekonfliktního řešení, následným přepočtem se agenti vrátili ke konfliktnímu stavu. Další studium tohoto problému by se mohlo zabývat nalezením vhodných závislostí, kdy se agent sám v průběhu argumentace rozhodne přepočítat stav svého grafu.

# Literatura


[1] PARSONS, S.; SIERRA, C.; JENNINGS, N.: Agents that Reason and Negotiate by Arguing. *Oxford University Press*, 1998.

[2] PARSONS, S.; WOOLDRIDGE, M.; AMGOUD, L.: An analysis of formal inter-agent dialogues. *ACM Digital Library*, 2002.

[3] SAMEK, J.; ZBOŘIL, F.: Hierarchical Model of Trust in Contexts.

[4] SIERRA, C.; JENNINGS, N.; NORIGEA, P.; aj.: A framework for argumentation-based negotiation.

[5] ZBOŘIL, F.: *Agentní a multiagentní systémy - studijní opora.* FIT VUT v Brně, 2006.