FeVaRFiStTr - Algorithmische Kommutativität zwischen Gewichteten Merkmal-Werte-Relationen und Endlichen Gewichteten Transduktoren in ihrer Funktion als Semantische Träger

Niclas Geiger, Markus Huber, Christian Kölbl,
Moritz Laudahn, Rupert Reutner-Hammelmeir, Frowin Ziegler

Universität Augsburg
christian.koelbl@informatik.uni-augsburg.de

Kurzfassung: Im folgenden Beitrag wird das Tool FeVaRFiStTr zur semantischen Verarbeitung von Dialogschritten mittels gewichteten Merkmal-Werte-Relationen und endlichen gewichteten Transduktoren unter Berücksichtigung deren algorithmischer Kommutativität vorgestellt. Das Tool ist der Beginn der Implementierung einer stetigen Forschung an dieser Thematik und soll bis zum Semantikflusssimulator weiterentwickelt werden.

1 Einleitung

In unserem aktuellen Projekt versuchen wir ein hierarchisches Sprachdialogsystem um eine semantische und pragmatische Ebene gemäß Abbildung 1 zu erweitern. Dabei gehen wir von einem begrenzten Weltmodell aus, d.h. das Dialogsystem ist lediglich in der Lage das zu verstehen, was in diesem Modell festgelegt ist.

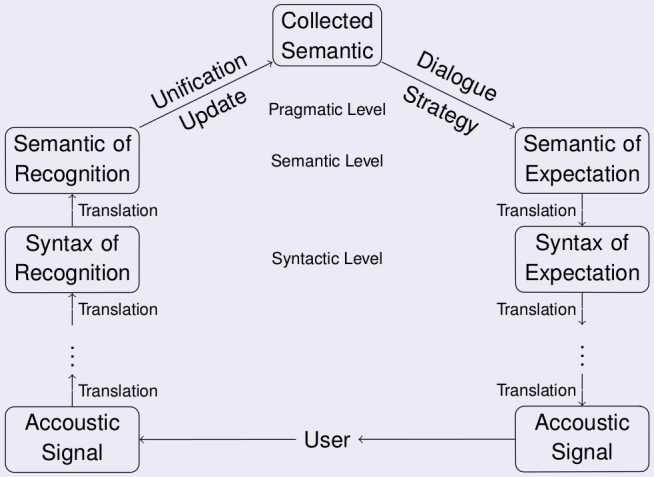

Abbildung 1 - Skizzenhafter hierarchischer Aufbau eines um Semantik und Pragmatik erweitertes Sprachdialogsystem.

Hierbei werden in jedem Dialogschritt die Ergebnisse der syntaktischen Ebene mittels sog. *Utterance-Meaning-Pairs* [8] auf alle möglichen Semantiken gemappt und anschließend gespeichert. Ein etablierter Ansatz besteht darin, alle relevante Information in einem *Informationszustand* zu speichern (siehe z. B. [6, 7]), und darin neu ankommende Information zu integrieren,

also ein *Information-State Update* (ISU) zu realisieren. Etwas formaler besteht der ISU-Ansatz aus

- einem Informationszustand, der Aspekte des gemeinsamen Kontexts zwischen System und Benutzer sowie weitere für den Dialog relevante Information enthält,
- einer Menge von möglichen Dialogschritten des Systems,
- einer Menge von Update-Regeln, die es erlauben, Information aus der Spracherkennung einzuarbeiten.

Im Anschluss daran wird mittels eines *Dialogue-Strategy-Controllers* der aktuelle Informationsstatus ausgewertet und die Semantik einer Nachfrage inklusive einer passenden Erwartungshaltung an den Benutzer generiert. Diese wird schließlich synthetisiert und an den Benutzer gerichtet und der nächste Dialogschritt beginnt.

Um dies realisieren zu können benötigt man auf semantischer und pragmatischer Ebene eine Datenstruktur, die in der Lage ist unsichere semantische Information abzubilden. Eine solche Datenstruktur nennen wir semantischen Träger. In [3] haben wir einen dynamischen semantischen Träger als Erweiterung von Attribut-Wert-Matrizen unter dem Begriff der gewichteten Merkmal-Werte-Relation (gMWR) eingeführt. Damit ist es möglich die Semantik des Gesagten abzubilden. Diese Information kann dann mittels eines eigens auf dieser Struktur definierten Update-Algorithmus [5, 2] in den ebenfalls als gMWR vorliegenden Informationsstatus eingebunden werden.

In den zwei Jahren danach haben wir darüber hinaus gemäß Abbildung 2 eine algorithmische Kommutativität[1] zwischen gMWRen und endlichen gewichteten Transduktoren (kurz FSTs für Finite State Transducers) nachgewiesen [4]. Hierdurch können auch FSTs als semantische Träger und somit zur Modellierung der Semantik und Pragmatik unseres Dialogsystems genutzt werden.

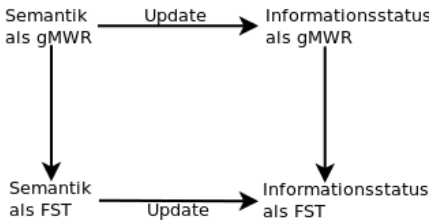

Abbildung 2 - Kommutativität zwischen gMWR und FST.

In diesem Jahr wollen wir vorstellen, wie die o.g. Ergebnisse in eine Anwendung mit dem Namen FeVaRFiStTr[2] geflossen sind und welche weiteren Pläne wir mit dieser verfolgen.

Im nächsten Abschnitt werden wir auf die Funktionalität des Tools genauer eingehen und anschließend beschreiben wir noch dessen Architektur. Im vierten Abschnitt soll mittels eines Use Cases die praktische Anwendung exemplarisch dargestellt werden. Der vorletzte Abschnitt enthält noch einen Ausblick auf die Funktionalitäten und Erweiterung der kommenden Versionen und schließen werden wir mit einigen Hinweisen zur Installtion.

2 Funktionalität

FeVaRFiStTr ist eine Eclipse[3]-Applikation, die derzeit noch hauptsächlich auf das Erstellen und Editieren von gewichteten Merkmal-Werte Relationen ausgelegt ist, allerdings bis zur Engine erweitert werden soll, welche dann die Simulation des Semantikflußes eines Dialogs ermöglicht.

Wie eingangs erwähnt ist die Thematik des Dialogsystems durch ein Weltmodell beschränkt. Dieses ist hier eine MWR[4], welche aus Aktionen und Datensätzen, die zu der Ausführung von

[1] Eine algorithmische Kommutativität lässt zu jeder Zeit eine Übersetzung zwischen FST und MWR zu, wobei die Kommutativität aber im mathematischen Sinne noch nicht bewiesen ist.

[2] Das Akronym steht für Feature-Values-Relation ↔ Finite-State-Transducer

[3] `http://www.eclipse.org/`

[4] Das Weltmodell besitzt keine Gewichte, da es lediglich die strukturellen Zusammenhänge erfasst

Abbildung 3 - Toolbar. (v.l.:Neue MWRen erstellen, Valdierung, Redo, Undo, Auswahl löschen, Informationen über das Tool, Speichern, Laden, Import von DB, Update durchführen, Screenshot, Export als FST, Zoom)

Aktionen benötigt werden, besteht. Derzeit ist es möglich besagtes Modell entweder frei Hand zu erstellen oder die Datensätze aus einer angebundenen SQL-Datenbank generieren zu lassen - die Aktionen müssen dennoch selbst erstellt und angebunden werden. Ein Validierungssystem stellt sicher, dass es sich hierbei um strukturell korrekte MWRen handelt (Abbildung 4).

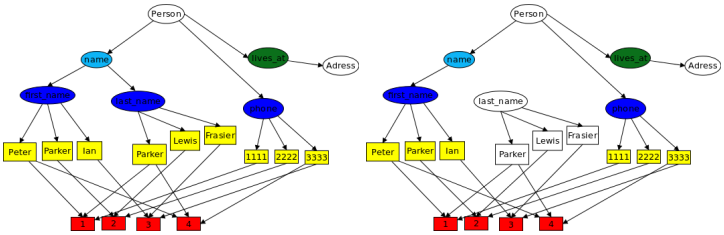

Abbildung 4 - Die rechte Graphik ist nicht valide und enthält neben den Wurzelknoten weitere farblose Elemente. Die linke ist korrekt.

Abbildung 5 zeigt das Interface des FeVaRFiStTr mit einem Bestandteil des für den Use Case in Abschnitt 4 benötigten Weltmodells (bei dem die Aktion *Call* mit einem Datensatz *Person* ausgeführt wird).

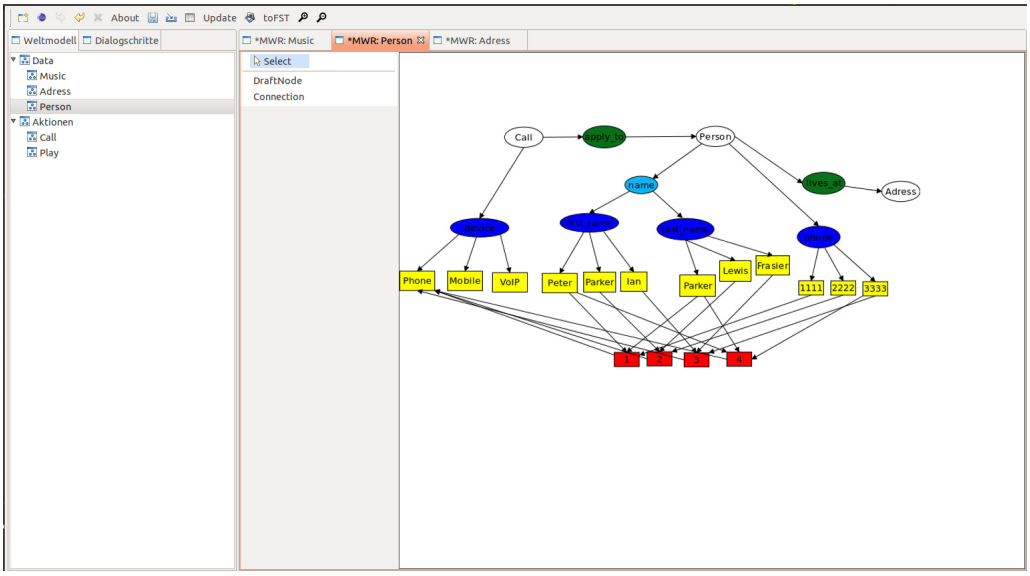

Abbildung 5 - Interface des FeVaRFiStTr.

Basierend auf diesem Weltmodell wird der sogenannte Informationsstatus (kurz SoIn für *State of Information*) als gewichtete MWR generiert, welcher in der Lage ist eingehende Information über die Welt zu speichern. Diese eingehende Information wird wiederum als gMWR(en) codiert und spiegelt die Semantik des Gesagten wider. Die Gesamtheit dieser gMWRen nennen wir *Horizont of Recognition*, oder kurz HoRs. Im Reiter *Dialogschritte* (Abbildung 6) kann man einen solchen Verlauf schrittweise simulieren.

Die generierten MWRen können nach einem eigens dafür angeleten Schema in einer XML-Datei gespeichert oder davon geladen werden. Der Screenshot Button ermöglicht den Export der erzeugten MWRen als Bilddatei[5]. Zu jeder Zeit ist außerdem ein Export der aktuellen MWRen als gewichteter Finite State Transducer in openfst-Code möglich.

3 Architektur

FeVaRFiStTr ist eine auf RCP[6] basierende Eclipse-Applikation. RCP-Applikationen nutzen lediglich den Kern von Eclipse - unabhängig der Entwicklungsumgebung, welche beispielsweise als *eine* mögliche RCP-Applikation gesehen werden kann. Das Tool ist in Java 6 geschrieben und kann demnächst unter `http://www.informatik.uni-augsburg.de/lehrstuehle/inf/mitarbeiter/koelbl/Forschung/`heruntergeladen werden. Der Download enthält außerdem ein vollständiges und aktuelles UML-Diagram der Implementierung im Violet[7]-Format. Der Code folgt dem Model-View-Controller-Prinzip und ist mittels Javadoc kommentiert. Als Datenbanksystem wird MySQL genutzt. Der Export der MWRen zu FSTs funktioniert über die Generierung von Symbol-Dateien, die in einem angepassten Skript mittels openfst[8] kompiliert, in eine Graphviz dot-Datei übersetzt und schließlich als Postscript-Graphik angezeigt werden.

4 Use Case

Eine derzeit typische Anwendung ist die Abbildung eines Dialogs bei dem sowohl Ambiguitäten aufgelöst werden müssen, wie auch Aktionswechsel stattfinden. Wir gehen weiter von einem aktionsorientierten Dialog aus, was bedeutet, dass wir alles Gesagte einem Dialog zuordnen solange keine neue Aktion genannt wurde. Ist dies der Fall, wird ein Stack von Aktionen angelegt, welcher gemäß einer geeigneten Strategie abgearbeitet wird.
Wir greifen hierfür das Beispiel eines solchen Dialogs aus [1] auf. Die zugrunde liegende Datenbak besteht aus drei primären Tabellen und einer Assoziationstabelle:

Person				
pID	first_name	last_name	phone	...
1	Peter	Parker	111	...
2	Parker	Lewis	222	...
3	Ian	Frasier	333	...
4	Peter	Parker	333	...
⋮	⋮	⋮	⋮	⋮

lives_at	
pID	aID
1	1
2	2
3	3
4	3
⋮	⋮

Address			
aID	number	street	...
1	742	Evergreen Terrace	...
2	112	Mulholland Drive	...
3	42	Abbey Road	...
⋮	⋮	...	...

Music			
mID	album	artist	...
1	At San Quentin	Johnny Cash	...
2	Ring Of Fire	Johnny Cash	...
3	Swing Easy	Frank Sinatra	...
⋮	⋮	⋮	⋮

[5] Alle hier im Artikel verwendeten Bilder entstammen dem Bildexport aus FeVaRFiStTr
[6] `http://wiki.eclipse.org/index.php/Rich_Client_Platform`
[7] `http://alexdp.free.fr/violetumleditor/page.php`
[8] `http://www.openfst.org/twiki/bin/view/FST/WebHome`

Die unterstützten Aktionen seien außerdem *Call* und *Play*. Die Aktion *Call* benötigt die Entität *Person* (vgl. Abbildung 5, wobei der Import aus der Datenbank noch um einen *semantischen Knoten* "name" erweitert wurde.) zu ihrer Ausführung und die Aktion *Play* die Entität *Music*. Die Entität *Adress* ist zwar keiner Aktion zugeordnet, kann aber zur Identifikation einer Person über deren gegenseitige Assoziation dienen wie wir in folgendem Dialog sehen werden:

1 : USER: I want to call Parker.
2 : SYSTEM: Is Parker the first name?
3 : USER: No, I mean Peter Parker.
4 : SYSTEM: Which Peter Parker do you want to call?
5 : USER: Change that terrible song to something from Johnny Cash.
6 : SYSTEM: Which album by Johnny Cash?
7 : USER: At San Quentin
<<system starts playing >>
8 : SYSTEM: Which Peter Parker do you want to call?
9 : USER: The one who lives at 742 Evergreen Terrace.
<<calling the selected partner >>

"Parker" stellt in 1 eine semantisch uneindeutige Aussage dar, da es gemäß der Datenbank drei mögliche Interpretationen gibt. Der Benutzer könnte "Parker Lewis" oder einen der beiden "Peter Parker" gemeint haben. Entsprechend werden drei gMWRen im ersten HoRs erzeugt die nun in die passenden MWRen des SoIns integriert werden müssen. An dieser Stelle sollte angemerkt werden, dass zu jeder möglichen Aktion-Datensatz-Kombination eine gMWR erzeugt wurde. Aufgrund der Struktur wird nun überprüft welches der *passende* ist (aufgrund der anfangs erwähnten Utterance-Meaning-Pairs und des Weltmodells kann an dieser Stelle nur Semantik stehen, die tatsächlich auch im System existiert). Diese Situation wird in Abbildung 6 dargestellt.

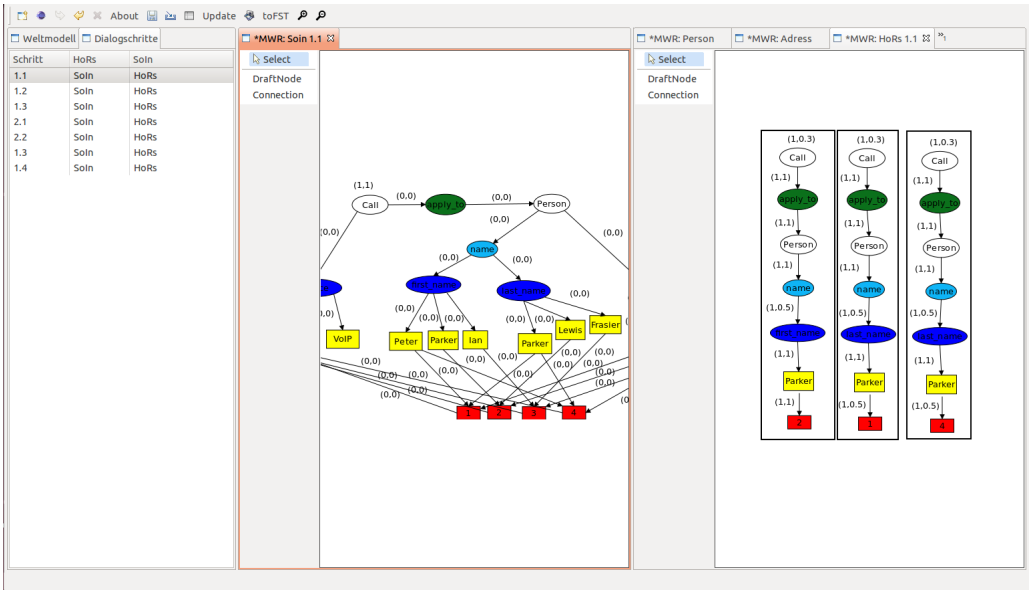

Abbildung 6 - Situation vor dem Update einer SoIn-MWR durch drei HoRs-MWRen

Alle drei MWRen des HoRs sind semantisch gleichwertig und müssen auch so behandelt werden. Das Resultat sind drei MWRen im nächsten SoIn nach dem Update (Abbildung 7).
Das Update arbeitet nach einem festen Algorithmus allerdings ist angedacht die Update-Funktion beliebig anpassbar zu lassen. Aktuell versuchen wir nachzuweisen, dass gewichtete MWRen analog zu FSTs über Halbringen definiert und somit noch flexibler gehandhabt werden können.

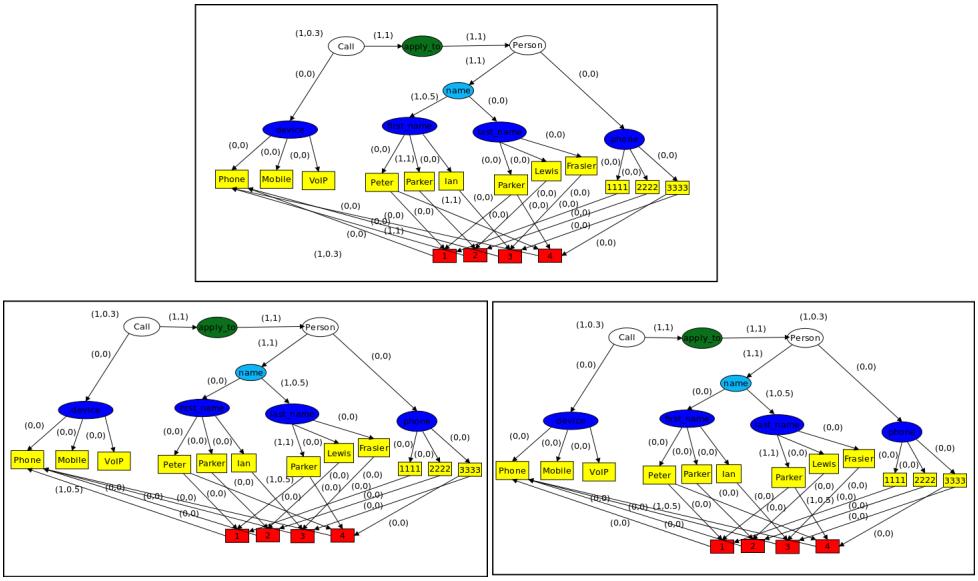

Abbildung 7 - Situation nach dem Update von Abbildung 6 (exportiert aus FeVaRFiStTr)

Die Erweiterung des Editors zur Engine beinhaltet außerdem die Implementierung eines sog. “Disco”[9] welcher in jedem Schritt die MWRen auswertet. In diesem Fall würde Disco feststellen, dass es einen Widerspruch gibt, welcher aufgelöst werden muss und würde entsprechend des Dialogverlaufs nachfragen. Derzeit muss dies aber noch händisch simuliert werden.

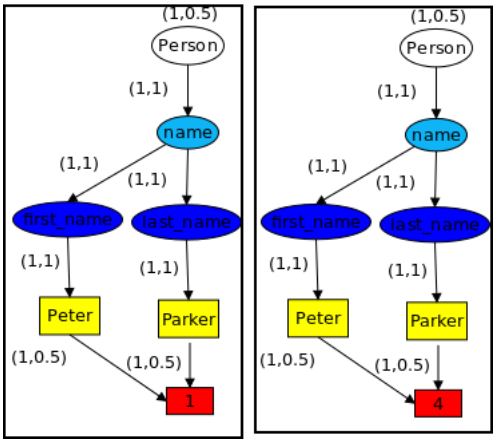

Abbildung 8 - Die Antwort “Peter Parker” (exportiert aus FeVaRFiStTr)

Der nächste Schritt disambiguiert zwar zwischen Vor- und Nachnamen, allerdings existieren nun zwei Peter Parker. Die gMWRen des HoRs (Abbildung 8) sind nun nicht mit der gMWR des SoIns welche Parker als Vornamen repräsentiert vereinbar. Auch beinhalten sie keine neue Aktion, folglich wird diese gMWR des SoIns beim zweiten Update verworfen und der SoIn 1.3 besteht lediglich aus zwei MWRen, welche wiederum über eine Nachfrage disambiguiert werden sollen.

Der Benutzer wechselt aber jetzt die Aktion und nachdem unser Dialogsystem aktionsorientiert arbeitet wird an dieser Stelle vom System festgestellt, dass es sich um keine vereinbare MWR mit dem aktuellen SoIn handelt und zudem noch eine andere Aktion aufgerufen wird. Unsere derzeit verfolgte LIFO[10]-Strategie erzeugt an dieser Stelle einen neuen SoIn 2.1. und verfolgt

[9] Dialogue Strategy Controller
[10] Last In First Out

nun die Ausführung dieser Aktion. Nach (erfolgreicher) Beendigung dieser wird wieder mit der vorherigen Aktion fortgefahren bis alle im Action-Stack vorhandenen Aktionen abgearbeitet oder vom Benutzer abgebrochen worden sind.

Zu jeder Zeit des Dialogs ist es möglich die aktuellen MWRen in einen FST gemäß [4] zu übersetzen.

Es bleibt noch anzumerken, dass die im Beispiel verwendeten Gewichte Testgewichte der Implementierung sind und nicht der Realität entsprechen. Zudem gehen wir davon aus, dass ausnahmslos das semantisch codiert wird, was auch gesagt wurde. Tatsächlich handelt es sich aber um eine Liste möglicher Erkennalternativen [9] unter denen auch unpassende zu finden sind.

5 Ausblick

Während sich die Funktionalität des Editors bisher lediglich auf die Modellierung der genannten semantischen Träger sowie deren Aktualisierung beschränkt, soll in der weiteren Entwicklung ein Simulationsmodus für komplette Dialoge hinzugefügt werden, um den Editor schließlich als Engine nutzen zu können. Im letzten Entwicklungsschritt soll so ein Semantikflusssimulator entstehen, der über eine geeignete API an beliebige Applikationen angebunden werden kann. Für diesen Ausbau wird allerdings noch eine Künstliche Intelligenz in Form eines Dialog-Strategie-Controllers benötigt, welche derzeit Gegenstand unserer aktuellen Forschung ist. Dieser hat dann die Aufgaben der Benutzeradaption, Disambiguierung anhand der Dialoghistorie, Generierung des nächsten Dialogschritts sowie der Konstruktion einer Erwarungshaltung des Dialogsystems gemäß des geplanten Dialogschritts.

Die nächsten Schritte umfassen die benutzerfreundlichere Gestaltung des Interfaces, Implementierung einer editierbaren Form für FSTs, Multilinguale Unterstützung, Anbindung des Tools als Open Source - Projekt an die Sourceforge-Community, Erzeugung von Datenbanktabellen aus MWRen, Anwendung auf größere Datenmengen, automatisierter Anordnungsalgorithmus uvm.

6 Installation

Der aktuelle Editor wurde unter Ubuntu 10.04 und 12.04 getestet.

Die Installation benötigt eine Eclipse-Installation mit den Erweiterungen Graphical Editing Framework GEF Examples, Graphical Editing Framework GEF, Equinox p2 RCP Management Facilities Source, Equinox p2 Provisioning for IDEs sowie Eclipse RCP. Anschließend kann der Code heruntergeladene Code wie gewohnt als bestehendes Projekt in Eclipse eingebunden werden. Zur Übersetzung von MWRen in FSTs wird außerdem eine Installation von openfst benötigt. Eine Anleitung zur Erstellung eines Skripts zur Automatisierung unter diversen Betriebssystemn liegt dem Download bei. Schließlich kann noch über die Settings eine MySQL-Datenbank angebunden werden.

Literatur

[1] G., W., H. M., K. C., L. R. und R. R. (Hrsg.): *Semantic dialogue modeling*, Lecture Notes in Computer Science. Springer, 2012. in print.

[2] HUBER, M.: *Semantische Informationsbearbeitung unter Berücksichtigung von Konfidenz-Zahlen*. Diplomarbeit in Mathematik. Katholische Universität Eichstätt-Ingolstadt, 2009.

[3] HUBER, M., C. KÖLBL, R. LORENZ, R. RÖMER und G. WIRSCHING: *Semantische Dialogmodellierung mit gewichteten Merkmal-Werte-Relationen*. In: HOFFMANN, R. (Hrsg.):

Elektronische Sprachsignalverarbeitung 2009. Tagungsband der 20. Konferenz. Dresden, 21. bis 23. September 2009, Bd. 53 d. Reihe *Studientexte zur Sprachkommunikation*, S. 25–32. TUDpress, Sep. 2009.

[4] HUBER, M., C. KÖLBL und G. WIRSCHING: *Gewichtete endliche Transduktoren als semantische Träger*. In: KRÖGER, B. und P. BIRKHOLZ (Hrsg.): *Elektronische Sprachsignalverarbeitung 2011. Tagungsband der 22. Konferenz. Aachen, 28. bis 30. September 2011*, Bd. 61 d. Reihe *Studientexte zur Sprachkommunikation*, S. 176–183. TUDpress, Sep. 2011.

[5] MATT, A.: *Unifikation semantischer Strukturen*. Diplomarbeit in Mathematik. Katholische Universität Eichstätt-Ingolstadt, 2009.

[6] TORBJÖRN LAGER, F. K.: *Implementing the Information-State Update Approach to Dialogue Management in a Slightly Extended SCXML*. S. 49–56, 2007.

[7] TRAUM, D. und S. LARSSON: *The Information State Approach to Dialogue Management*. In: *Current and New Directions in Discourse and Dialogue*, S. 325–353. 2003.

[8] WIRSCHING, G. und C. KÖLBL: *Language Modeling with Utterance-Meaning Pairs*. Techn. Ber., Angewandte Informatik, Universität Augsburg, 2011.

[9] WIRSCHING, G., C. KÖLBL und M. HUBER: *Zur Logik von Bestenlisten in der Dialogmodellierung*. In: KRÖGER, B. und P. BIRKHOLZ (Hrsg.): *Elektronische Sprachsignalverarbeitung 2011. Tagungsband der 22. Konferenz. Aachen, 28. bis 30. September 2011*, Bd. 61 d. Reihe *Studientexte zur Sprachkommunikation*, S. 309–316. TUDpress, Sep. 2011.