

Hierarchical Joint Learning for Natural Language Generation



Dissertation zur Erlangung der Doktorwürde
durch den Promotionsausschuss Dr. Phil.
der Universität Bremen

Nina Saskia Dethlefs

Bremen, den 25. Oktober, 2013

Zusammenfassung

Die größte Schwierigkeit natürlichsprachlicher Generierungssysteme ist häufig die Selektion der treffendsten und authentischsten Äußerung aus einer Vielzahl möglicher Formulierungen. Der Grund hierfür ist in der Regel, dass die Qualität einer Äußerung stark situationsabhängig ist. Erst durch das komplexe Zusammenspiel aus physikalischer Umgebung, pragmatischen Begebenheiten, dem jeweiligen Adressaten und der vorliegenden Interaktionsgeschichte ergibt sich der Kontext, der eine Äußerung geeignet oder ungeeignet erscheinen lässt.

In der Vergangenheit haben sich Entwickler natürlichsprachlicher Generierungssysteme wiederholt dem Pipeline-Modell bedient. Dieses gliedert den Generierungsprozess in drei aufeinanderfolgende Arbeitsschritte. Zunächst wird in der *Inhaltsauswahl* die Semantik der Äußerung bestimmt, die eine erste logische Form für die weitere Generierung festlegt. Im zweiten Schritt des Modells, der *Äußerungsplanung*, wird diese Form in Subäußerungen untergliedert. Schließlich folgt die *Oberflächengenerierung*, in der eine Wortsequenz ausgewählt wird, welche nachfolgend dem Benutzer präsentiert werden kann.

Alle drei Arbeitsschritte haben einen maßgeblichen Einfluss auf die durch die jeweilige Situation bestimmte Effizienz der letztlich gewählten Äußerung und die kognitive Belastung des Benutzers. In der Praxis wird das sequentielle Pipeline-Modell allerdings nicht den Wechselwirkungen gerecht, die zwischen den einzelnen Arbeitsschritten bestehen.

In dieser Dissertation wird ein Modell der zusammenhängenden Optimierung für situierte, natürlichsprachliche Generierung entwickelt,

das auf der computergestützen Theorie des *hierarchischen, verstärkenden Lernens* basiert und zudem mit grafischen Modellen verbunden wird. Dieses zusammenhängend-optimierte Modell *lernt* die beste Äußerung in einer gegebenen Situation durch eine Versuch- und Irrtum-Suche. Es betrachtet linguistische Entscheidungen nicht singulär, sondern berücksichtigt die bestehenden Interdependenzen und Reziprozitäten zwischen ihnen. So generiert das zusammenhängend-optimierte Modell deutlich kontext-sensitivere Äußerungen als Modelle, die diese Wechselwirkungen außer Acht lassen. Durch die Einführung eines *hierarchischen Informationsstandes* wird die Natürlichkeit der Sprache gesteigert. Hierdurch wird einerseits die systematische Spezifizierung von Vorwissen, andererseits die Integration menschlicher Präferenzen bei der Inhaltsauswahl ermöglicht.

In das Lernmodell werden als grafische Modelle Markov Modelle und Bayessche Netze integriert, um unterschiedliche linguistische Oberflächen zu repräsentieren und das Verhältnis zwischen Variation der Äußerung und Anpassung an den Benutzer zu optimieren.

Ergebnisse einer Evaluationsstudie belegen, dass das hierarchische Lernmodell eine zuverlässige Generierungsstrategie lernt, die sich neuen Begebenheiten und Benutzern anpasst und die flexible und erfolgreiche Interaktionen ermöglicht. In einem Vergleich führte die zusammenhängend-optimierte Strategie zu einer deutlich höheren Benutzerzufriedenheit und Erfolgsrate als eine isoliert-optimierte Strategie. Weiterhin nahmen die Benutzer die Äußerungen der zusammenhängend-optimierten Strategie als deutlich natürlicher wahr. Um die Domänenunabhängigkeit und das Generalisierungspotential des entwickelten Modells zu veranschaulichen, wird eine zusätzliche Studie präsentiert, die das Lernmodell auf eine neue, jedoch verwandte, Domäne überträgt: die Generierung von Routenbeschreibungen in einem realen Navigationsszenario unter Benutzung eines situierten Dialogsystems für innerräumliche Navigation. Die Ergebnisse dieser Studie bestätigen, dass sich die gelernte Generierungsstrategie mit begren-

ztem Aufwand auf neue Domänen übertragen lässt und zu hoher Benutzerfreundlichkeit und Erfolg beiträgt.

Abstract

Natural Language Generation (NLG) systems typically face an uncertainty regarding the best utterance to communicate to a user in a given context given that the effect of a single utterance depends crucially on the interplay between its physical environment, pragmatic circumstances, addressee and interaction history. NLG system designers have traditionally used a pipeline architecture to divide the generation process into the distinct stages of *content selection*, *utterance planning* and *surface realisation* to choose the semantics, organisation and realisation of an utterance. Unfortunately, this sequential model does not account for the interdependencies that exist among these stages, which in practice has been manifest in inefficient instruction giving and an increased cognitive load for the user.

This thesis will advocate a *joint optimisation* framework for situated NLG that is based on *Hierarchical Reinforcement Learning* combined with *graphical models* and will *learn* the best utterance for a given context by optimising its behaviour through a trial and error search. The joint model considers decisions at different NLG stages in interdependence with each other and thereby produces more context-sensitive utterances than is possible when considering decisions in isolation. To enhance the human-likeness of the model, two augmentations will be made. We will introduce the notion of a *Hierarchical Information State* to support the systematic pre-specification of prior knowledge and human preferences for content selection. Graphical models—Hidden Markov Models and Bayesian Networks—will then be integrated as generation spaces to encourage natural surface realisation by balancing the proportion of alignment and variation.

Results from a human evaluation study show that the hierarchical learning agent learns a robust generation policy that adapts to new circumstances and users flexibly leading to smooth and successful interactions. In terms of the comparison between a joint and an isolated optimisation, results indicate that a jointly optimised system achieves higher user satisfaction and task success and is better perceived by human users than its isolated counterpart. To demonstrate the domain-independence and generalisabilty of the hierarchical joint optimisation framework, an additional study will be presented that transfers the model to a new, but related, domain: the generation of route instructions in a real navigation scenario using a situated dialogue system for indoor navigation. Results confirm that the NLG policy can be applied to new domains with limited effort and contribute to high task success and user satisfaction.

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und ist auch noch nicht veröffentlicht.

Nina Dethlefs,
Bremen, im April 2012.

Veröffentlichungen

Teiler dieser Dissertation wurden bereits in Konferenzen oder Workshops vorgestellt. Eine komplette Liste meiner bisherigen Veröffentlichungen ist die Folgende.

Journal articles and book chapters

Nina Dethlefs (to appear) Context-Sensitive Natural Language Generation: From Knowledge-Driven to Data-Driven Approaches. *Language and Linguistics Com-pass*.

Nina Dethlefs and Heriberto Cuayáhuitl (to appear) Hierarchical Reinforcement Learning for Situated Language Generation. *Natural Language Engineering*. Cambridge University Press.

Nina Dethlefs and Heriberto Cuayáhuitl (to appear) A Joint Learning Approach for Situated Language Generation. In: Stent, A. and S. Bangalore (eds.) *Natural Language Generation in Interactive Systems*. Cambridge, England: Cambridge University Press.

Heriberto Cuayáhuitl and Nina Dethlefs (2011) Spatially-aware Dialogue Control Using Hierarchical Reinforcement Learning. *ACM Transactions on Speech and Language Processing (Special Issue on Machine Learning for Robust and Adaptive Spoken Dialogue Systems)* 7:3, pp. 5:1-5:26.

Nina Dethlefs, Yunhui Wu, Aisan Kazerani and Stephan Winter (2011) Generation of Adaptive Route Descriptions in Urban Environments. *Spatial Cognition and Computation* 11:2, pp. 153-177.

Thora Tenbrink, Robert Ross, Kavita Thomas, Nina Dethlefs and Elena Andonova (2010) Route instructions in map-based human-human and human-computer dialogue: a comparative analysis. *Journal of Visual Languages and Computing* 21:5, pp. 292-309.

Conference and Workshop contributions

Nina Dethlefs and Heriberto Cuayáhuitl (2012) Comparing HMMs and Bayesian Networks for Surface Realisation. To appear in *Proceedings of the 12th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Montréal, Canada.

Nina Dethlefs and Heriberto Cuayáhuitl (2011) Combining Hierarchical Reinforcement Learning and Bayesian Networks for Natural Language Generation in Situated Dialogue. To appear in *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France.

Heriberto Cuayáhuitl and Nina Dethlefs (2011) Optimizing Situated Dialogue Management in Unknown Environments. In *Proceedings of INTERSPEECH*, Florence, Italy.

Nina Dethlefs, Heriberto Cuayáhuitl and Jette Viethen (2011) Optimising Natural Language Generation Decision Making for Situated Dialogue. In *Proceedings of the 12th Annual Meeting on Discourse and Dialogue (SIGdial)*, Portland, OR, USA.

Nina Dethlefs and Heriberto Cuayáhuitl (2011) Hierarchical Reinforcement Learning and Hidden Markov Models for Task-Oriented Natural Language Generation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, Portland, OR, USA.

Heriberto Cuayáhuitl, Nina Dethlefs, Lutz Frommberger, Kai-Florian Richter and John Bateman (2010). Generating Adaptive Route Instructions Using Hierarchical Reinforcement Learning. In *Proceedings of the International Conference on Spatial Cognition (Spatial Cognition VII)*, Portland OR, USA.

Nina Dethlefs and Heriberto Cuayáhuitl (2010) Hierarchical Reinforcement Learning for Adaptive Text Generation. In *Proceedings of the International Conference*

on Natural Language Generation (INLG), Dublin, Ireland.

Nina Dethlefs, Heriberto Cuayáhuitl, Kai-Florian Richter, Elena Andonova and John Bateman (2010) Evaluating Task Success in a Dialogue System for Indoor Navigation. In *Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*, Poznan, Poland.

Heriberto Cuayáhuitl, Nina Dethlefs, Kai-Florian Richter, Thora Tenbrink and John Bateman (2010) A Dialogue System for Indoor Wayfinding Using Text-Based Natural Language. *International Journal of Computational Linguistics and Applications*, 1 (1-2), pp. 285-304. Poster presented at the 11th Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2010).

Other Publications

Nina Dethlefs (2011) The Bremen System for the GIVE-2.5 Challenge. Poster paper to appear in *Proceedings of the European Workshop on Natural Language Generation (ENLG)*, Nancy, France.

Nina Dethlefs (2011) Position Paper in the *Young Researchers Round Table on Spoken Dialogue Systems (YRRSDS)*. Portland, Oregon, USA.

Niels Schütte and Nina Dethlefs (2010) The Dublin-Bremen System for the GIVE-2 Challenge. Poster paper in the *GIVE-2 results poster session at INLG*, Dublin, Ireland.

Acknowledgements

As usual, this thesis would not exist in its current form without the help and support of many people. First of all, I would like to thank my supervisor John Bateman for introducing me to linguistics in my first semester at university. His course aroused my enthusiasm for a subject previously unknown to me, and made me realise that there are few things more exciting than the study of human language and its computational modelling. Most of all, though, I thank him for giving me the opportunity to take on a PhD in Computational Linguistics at a time when my enthusiasm was much bigger than my technical skills. Second, Heriberto Cuayáhuitl taught me what I needed to know to make it happen. To him, I am grateful for introducing me to the world of machine learning, for sharing so many ideas and discussions, saving me from a thousand stupid mistakes, being the best person to work with, always going the extra mile, and being cheerful in times of stress. Further, thanks to Michael Strube and David Schlangen for agreeing to review my thesis and making very good suggestions on how to improve it. I hope that I did them justice. Many thanks also to my examiners, Rainer Malaka and Lutz Frommberger.

Several people helped to make life in Bremen so much fun: Ann-Kathrin made sure I didn't miss out on Breminale, Viertelfest or Aqua-Fitness. With Moni I enjoyed many Thursday evenings over wine and chocolate watching the models walk. And with Martina I shared so many stories over the years, at some point a flat, and an infamous bubble tea... Thanks for being such a gifted maths tutor as well! I have enjoyed working with my colleagues Desi Zhekova (it was good fun sharing an office, too!), Dimitra Anastasiou, Ken (Cui)

Jian, and for a short time Vivien Mast, Daniel Vale and Fazleh Elahi. I also learned things from Thora Tenbrink and Kerstin Fischer, when I was working with them as a student assistant. Niels Schütte worked with me on the final steps towards the PhD and has since remained a ‘partner in crime’ working on similar things and sharing amazing videos of scuba diving and the idea for an ingenious mouse trap. 

During my PhD, I did two research visits to Australia. Stephan Winter made me feel very welcome when I was visiting his group in Melbourne. From him, Elina (Yunhui) Wu and Aisan Kazerani I learned many things about route generation which have influenced this thesis in interesting ways. Robert Dale let me visit his group at Macquarie University at a very nice time of year in Sydney, and Jette Viethen discussed with me the psycholinguistics of referring expressions, when she was very busy finishing off her own PhD. Some of her advice has also entered this thesis. A few other people deserve a mention as well. Alexandre Denis and Kristina Striegnitz were endlessly patient while I was putting together my first GIVE system and helped with many compilation errors and bug fixes. Kristina also helped me make sense of the GIVE evaluation data. Alexander Koller and Konstantina Garoufi invited me to visit them in Potsdam and gave me a fantastic preparation for my viva; and Nadine Hagemann was an invaluable help in corpus annotation. Finally, thanks to Helen Hastie and Oliver Lemon for giving me an exciting postdoc job and to my colleagues at Heriot-Watt University for making it so easy to join their group.

Last but not least, my family have supported me throughout this endeavour in countless ways and I’m so glad I have you all. Especially thanks to Mama for making home such a warm and cosy place to return to, Papa for helping me move places, however far it has been, and Julia for telling stories of those travels around the world that I haven’t yet been on. And to my grandparents for always taking an interest in what I do. Finally, my baby son Tom and his Papa make every day a little brighter; you’re my whole world. 

Sometimes, said Pooh, the smallest things take up the most room in your heart. –A.A.Milne

For Tom. ♡

Glossary

AI	Artificial Intelligence
BN	Bayesian Network
BTS	Binary Task Success
CAS	Constituent Alignment Score
CFG	Context-Free Grammar
CS	Content Selection
DBN	Dynamic Bayesian Network
GIVE	Generating Instructions in Virtual Environments
GTS	Graded Task Success
HIS	Hierarchical Information State
HMM	Hidden Markov Model
HRL	Hierarchical Reinforcement Learning
HSMQ	Hierarchical Semi-Markov Q-Learning
IS	Information State
KLD	Kullback-Leibler Divergence
MDP	Markov Decision Process
MLE	Maximum Likelihood Estimation
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
PCFG	Probabilistic Context-Free Grammar
pCRU	Probabilistic Context-Free Underspecification
REG	Referring Expression Generation
RL	Reinforcement Learning
SMDP	Semi-Markov Decision Process
SR	Surface Realisation
UP	Utterance Planning

Contents

Contents	xv
List of Figures	xxi
List of Tables	xxix
1 Introduction	1
1.1 Context-Sensitive Situated NLG	1
1.2 Two Architectures for NLG	3
1.3 Outline of the Thesis	9
2 Related Work	11
2.1 Introduction	11
2.2 Context-Sensitive Language Generation	12
2.2.1 Rule-Based Approaches to Generation	12
2.2.2 Generation As Planning	15
2.2.3 Trainable Generation	17
2.2.4 Other Approaches	20
2.3 Reinforcement Learning for NLG	21
2.4 Joint Treatment of Subtasks	25
2.5 Graphical Models for Natural Language Generation	30
2.6 Conclusion	32
3 Hierarchical Reinforcement Learning for NLG	35
3.1 Introduction	35
3.2 Reinforcement Learning	37
3.2.1 The Markov Decision Process	39
3.2.2 Policy Learning	41
3.2.2.1 Estimating Value Functions	42
3.2.2.2 The Q-Learning Algorithm	45
3.2.3 An MDP for Referring Expression Generation	47
3.3 Hierarchical Reinforcement Learning	49

CONTENTS

3.3.1	The Semi-Markov Decision Process	50
3.3.2	Policy Learning	52
3.3.2.1	Estimating Value Functions	52
3.3.2.2	The HSMQ-Learning Algorithm	54
3.3.3	An SMDP for Referring Expression Generation	56
3.4	A Joint Learning Agent for Situated Interaction	57
3.4.1	The Domain: Generating Instructions in Virtual Environments (GIVE)	59
3.4.2	The GIVE-2 Corpus	60
3.4.3	Corpus Annotation	60
3.4.4	Hierarchy of Learning Agents	64
3.5	Experimental Setting	65
3.5.1	The Simulated Environment	65
3.5.2	A Data-Driven Reward Function	68
3.5.2.1	A Wayfinding Study: Experimental Setting . . .	68
3.5.2.2	Graded Task Success	69
3.5.2.3	Objective and Subjective Measures	70
3.5.2.4	Correlation Analysis	71
3.5.2.5	Estimating a Performance Function	72
3.5.3	Training Parameters	74
3.6	Experimental Results	75
3.7	Conclusion	78
4	A Hierarchical Information State for Constrained Learning	81
4.1	Introduction	81
4.2	Content Selection for Situated Interaction	85
4.2.1	Decision Trees for Content Selection	86
4.2.1.1	Learnt Rules for Navigation Instructions	88
4.2.1.2	Learnt Rules for Referring Expressions	89
4.2.2	Consistency and Alignment in Human Data	89
4.3	Information State	91
4.3.1	Informational Components	91
4.3.2	Formal Representations	92

CONTENTS

4.3.3	Generation Moves	92
4.3.4	Update Rules	92
4.3.5	Update Strategy	93
4.3.6	Example of an Information State for NLG	93
4.4	Combining Hierarchical RL with a Hierarchical Information State	94
4.4.1	The Semi-Markov Decision Process Using an Information State	95
4.4.2	The HSMQ-Learning Algorithm Using Constrained Actions	96
4.5	Experimental Setting	97
4.5.1	A Reward Function for Consistency	97
4.5.2	Training Parameters	98
4.6	Experimental Results	98
4.6.1	Simulation-Based Results	99
4.6.2	Human Rating Study	100
4.7	Conclusion	102
5	Graphical Models for Surface Realisation	105
5.1	Introduction	105
5.2	Variation and Alignment in Human Data	106
5.2.1	Variation and Alignment in the GIVE Corpus	109
5.2.2	A Constituent Alignment Score	111
5.3	Representing Generation Spaces as Graphical Models	112
5.3.1	(Probabilistic) Context-Free Grammars	112
5.3.2	Hidden Markov Models	114
5.3.3	Bayesian Networks	118
5.3.4	Comparison of Graphical Models	122
5.4	Experimental Setting	124
5.4.1	Integrating Surface Realisation: A Three-Dimensional Reward Function for Situated NLG	124
5.4.1.1	Dimension 1: User Satisfaction	124
5.4.1.2	Dimension 2: Naturalness	125
5.4.1.3	Dimension 3: Balancing Alignment and Variation	125
5.4.1.4	Bringing All Dimensions Together	127

CONTENTS

5.5	Experimental Results	129
5.5.1	Simulation-Based Results	129
5.5.1.1	Comparison of Policies Using Graphical Models .	129
5.5.1.2	Effects of Graphical Models for the Joint Optimisation	131
5.5.2	Similarity with Human Authors	134
5.6	Conclusion	136
6	Evaluation	139
6.1	Introduction	139
6.2	Experimental Setting for Navigation in Virtual Environments . .	141
6.2.1	Experimental Methodology	141
6.2.2	Experimental Setup	144
6.2.2.1	Training Worlds	146
6.2.2.2	Evaluation Worlds	148
6.2.3	Experimental Results	150
6.2.3.1	Objective Metrics	151
6.2.3.2	Subjective Metrics	156
6.2.3.3	Comparison with Systems from the GIVE Challenge	162
6.2.3.4	Discussion of Results	168
6.3	Experimental Setting for Navigation in Real Environments . . .	171
6.3.1	Experimental Methodology	172
6.3.2	Experimental Setup	173
6.3.3	Experimental Results	176
6.3.3.1	Objective Metrics	176
6.3.3.2	Subjective Metrics	178
6.3.3.3	Discussion of Results	180
6.4	Conclusion	181
7	Conclusions and Future Research	185
7.1	Contributions and Findings	185
7.2	Future Directions	190
	Appendix A	195

CONTENTS

Appendix B	211
Appendix C	235
References	241

CONTENTS

List of Figures

1.1	A traditional pipeline architecture for NLG systems using sequential decision making for the core tasks of content selection, utterance planning and surface realisation (adapted roughly from [Reiter and Dale, 1997, 2000])	5
1.2	An architecture for NLG systems that treats the core tasks of content selection, utterance planning and surface realisation in a joint module that is subject to optimisation. The framework developed in this thesis is based on a joint architecture using hierarchical reinforcement learning.	6
1.3	Example scenes with alternative instructions (some more felicitous than others in their context) from the GIVE environment [Byron et al., 2009].	7
3.1	Illustration of the <i>curse of dimensionality</i> problem that reinforcement learning agents face. An agent's state space grows exponentially according to the number of variables taken into account (here plotted in log space).	36
3.2	Illustration of the agent-environment interaction (adapted from Sutton and Barto [1998], page 52).	42
3.3	Example scenes for referring expression generation: scenes contain a referent (circled) that needs to be uniquely identified with respect to its distractors (other buttons) and landmarks in the environment.	47
3.4	State and action sets for a flat reinforcement learning agent for referring expression generation. The state-action space of this agent is $2^2 \times 3^4 \times 4^2 \times 8 = 41472$	48
3.5	Dynamics of a Semi-Markov Decision Process or SMDP [Cuayáhuitl, 2009], Chapter 3. State transitions and rewards depend on the number of time steps, denoted τ , actions take to complete.	52

LIST OF FIGURES

3.6	A hierarchy of Semi-MDPs where parent subtasks can call child subtasks and arrows indicate the flow of execution. The indices i and j serve only to uniquely identify an agent in the hierarchy, they do not specify an execution sequence (which is subject to optimisation).	53
3.7	Flat and hierarchical state-action spaces for a simple learning agent for referring expression generation. Whilst the size of the flat agent has 41472 state-actions, the size of the hierarchical agent has only 420 state-actions. In the hierarchical case, a parent subtask can invoke child subtasks. When they terminate, control is returned to the caller.	58
3.8	Examples of instructions which can be categorised as high-level, low-level and mixed instruction mode (all describing the same situation) taken from the GIVE corpus. The arrows in the maps on the left show the route segment that is described in each instruction. The instruction follower's initial position is indicated by the person in the lower-left room.	61
3.9	Hierarchy of learning agents for content selection (solid lines), utterance planning (dashed lines) and surface realisation (dotted lines) of navigation and referring expression generation in situated scenarios. The arrows indicate the flow of control as it is passed down from parents to child agents. The agents are indexed by their policies $\pi_0^0 \dots \pi_5^3$ for SMDPs $M_0^0 \dots M_5^3$	63
3.10	Illustration of the training worlds that are the basis of the simulated environment. All worlds require skills for navigation and disambiguation at a medium level of difficulty.	66
3.11	Performance (in terms of average reward) of jointly learnt policies in comparison with policies learnt in isolation and several intermediate variants.	74
4.1	Example of a referring expression generation situation in the GIVE world where more than one description of the target referent (circled) is possible.	82

LIST OF FIGURES

4.2 Example of a decision tree concerned with whether or not the colour of the referent should be included in a referring expression. It appears that whenever an utterance is not a repair (e.g. a rephrase or repetition) of the previous utterance, the colour should be included.	88
4.3 A constrained Semi-Markov Decision Process for NLG, where I_t represents the information state at time t , s_t represents a knowledge-compact state, a_t is a generation action (primitive or composite) available in s_t , and r_t is a numerical reward for choosing action a_t . Note that the information state constrains the actions available per state, i.e. action a_t is selected if $a_t \in A \cap I$	96
4.4 Learning curves indicating the rewards (averaged over 10 runs) that the fully-learnt (Without constraints) and the semi-learnt (With constraints) agents received over 150 thousand learning episodes. While both agents learn equivalent policies, the semi-learnt agent learns an optimal policy earlier.	99
4.5 Example of a spatial scene matched with one learnt (a), one deterministic (b) and one human (c) instruction. Participants were asked to rate each instruction for its appropriateness and circle the intended referent.	101
5.1 Extract from a dialogue of the maze protocols exemplifying two cases of alignment (shown in blue). In the first case, ‘extreme right’, speaker B aligns lexically with speaker A. In the second example, ‘right indicator’, speaker B self-aligns. The example was adapted from Pickering and Garrod [2006] , p. 5–6.	108
5.2 Example PCFG for destination instructions. Nonterminal symbols represent semantic constituents and terminal symbols represent possible realisations. Probabilities of occurrence for individual surface forms are given behind the surface form, the probability of expansion for each rule is given in bold-face.	114

LIST OF FIGURES

5.3 Example Hidden Markov Models for predicting weather events (Ju-rafsky and Martin [2009] , Chapter 6) on the left-hand side, including transition and emission probabilities, and for Part-Of-Speech Tagging for the sentence ‘ <i>You turn left</i> ’	116
5.4 Example trellis for an HMM representing the generation space of destination instructions. The observation sequence corresponds to those semantic concepts that need to be realised. States correspond to possible surface realisations of the semantic concept. Note that only a subset of all states and transitions are shown. Dashed arrows correspond to sequences that occur in the data.	117
5.5 Example Bayesian Network for predicting whether the family is home or not, adapted from Charniak [1991] , including the conditional probability tables over random variables. The nodes (random variables) denote states of affairs and the arcs (uni-directional) causal connections.	119
5.6 Example Bayesian Network representing the generation space of destination instructions. Random variables correspond to semantic concepts and their values correspond to possible surface realisations of the semantic concept.	121
5.7 The hierarchy of learning agents with alternative graphical models shown in the bottom. They inform the surface realisation agents $M_{0..5}^3$ by providing feedback for the surface realisation choices in the form of rewards. Note that only one type of graphical model is used at a time.	126
5.8 The probability density function from which rewards are sampled to inform the agent’s learning process with respect to optimising the proportion of alignment and variation. The x-axis corresponds to different <i>constituent alignment scores</i> and the y-axis corresponds to probabilities assigned as rewards.	128

LIST OF FIGURES

5.9	Results in terms of average rewards for the joint learning framework including graphical models for surface realisation. In comparison to the plots in Figure 3.11 in Chapter 3, we can see an increase in performance due to the integration of graphical models. Bayesian Networks and Hidden Markov Models achieve the best performance, followed by PCFGs and CFGs.	131
5.10	Results in terms of average rewards that different graphical models achieve in isolation of the joint learning framework. We can see that Bayesian Networks reach the same performance within and without the joint framework, but the performance of Hidden Markov Models deteriorates when considered in isolation. PCFGs and CFGs reach the same (suboptimal) performance as before. . .	133
6.1	Photos of the experimental setting for the GIVE evaluation. Left: a participant is playing. Right: the computer that participants played on.	141
6.2	Illustration of training world 1. The focus here is on navigation of a low to medium level of difficulty and referring expressions that require a limited amount of disambiguation.	145
6.3	Illustration of training world 2. The challenge of this world is represented by its large rooms, which require high-level navigation skills in certain situations (such as crossing a large room or referring to a particular corner).	146
6.4	Illustration of training world 3. The challenge here lies in guiding the user successfully through a number of small rooms that are connected by a difficult junction in which a user can quickly lose orientation.	147
6.5	Illustration of evaluation world 1. This evaluation world was designed to be similar to the training worlds. It requires navigation with a medium degree of difficulty and limited disambiguation with respect to referring expressions.	148

LIST OF FIGURES

6.6	Illustration of evaluation world 2. In this world, the focus is on referring expression generation. A large number of same-coloured buttons are located close to each other in different spatial arrangements so that disambiguation becomes a challenge.	149
6.7	Illustration of evaluation world 3. This world requires sophisticated navigation skills in all rooms (the green corridor in which users can quickly lose orientation, the room with the red tiles where any wrong step may cause the alarm to be triggered and the room with the plants where specific references to doors need to be given).	150
6.8	Bar plots illustrating the comparison of the joint policy against the isolated policy in terms of binary task success and 4-valued and 3-valued graded task success. The joint policy outperforms the isolated policy according to all metrics indicating that joint interactions are more successful and smooth than their isolated counterparts and encounter fewer problems.	153
6.9	Box plots illustrating the user satisfaction ratings for the isolated policy (on the left) and for the joint policy (on the right). The isolated data show a trend towards more negative ratings than the joint policy (metrics Q1, Q4 and Q10) and the joint data show a trend towards more positive ratings than the isolated policy (metrics Q5, Q8 and Q9).	159
6.10	Illustration of two factors that explain 65% of the variability found in subjective user ratings: ‘usability’ and ‘pace’.	161
6.11	Evaluation worlds of the GIVE-2 challenge that was run in 2010 [Koller et al., 2010]. Again, World 1 was designed to be most similar to the training worlds by representing a medium degree of difficulty in navigation and referring expression generation. World 2 contained more complex referring expression generation scenarios and World 3 was challenging in terms of navigation.	164
6.12	Two example situations (unseen from training) from evaluation world 3. On the left, the system generates an ambiguous reference to a door. On the right, it generates a referring expression combined with a warning to not step on any red tile.	170

LIST OF FIGURES

- 6.13 Map of the navigation environment for the in-situ evaluation in a real setting. The six destinations that participants had to navigate to are shown as destination (a)–(f) on the map. The order of tasks was randomised. 174
- 6.14 The Android application that participants used to navigate to a set of six different destinations. On the left: the simulated phone used for development. On the right: the actual phone used during the evaluation with arrows pointing to buttons or text fields together with the description of their purpose. All experiments were run on the HTC Desire S mobile phone using Android 2.3. 175
- 6.15 Bar plots illustrating the binary and graded task success for the in-situ wayfinding scenario evaluated in this thesis in comparison to the baseline in-advance wayfinding scenario reported in Cuayáhuitl and Dethlefs [2011a]. The first two metrics, binary task success and 4-valued graded task success, seem to suggest that users are more successful in the in-situ scenario. The 3-valued graded task success metric in contrast points in the opposite direction suggesting that users encounter less problems in the in-advance scenario. 177
- 6.16 Bar plots comparing the average time taken by users to complete a wayfinding task in the in-situ scenario evaluated in this thesis (on the right) and in the baseline in-advance scenario (on the left) [Cuayáhuitl and Dethlefs, 2011a]. Interactions are visibly shorter and more efficient in the in-situ scenario. The difference corresponds to 41 seconds on average. 178

LIST OF FIGURES

6.17 Box plots illustrating the user satisfaction ratings for the baseline in-advance scenario [Cuayáhuitl and Dethlefs, 2011a] (on the left) and the in-situ scenario evaluated in this thesis (on the right). Users preferred the interaction pace (Q4) in the in-situ scenario and expect to use the system more often in the future (Q7) than the in-advance system. On the other hand, they indicate that they knew less often what to say (Q5) to the system in the in-situ setting than in the in-advance setting. This may be due to the fact that interaction in the in-situ scenario was controlled predominantly by button presses.	180
6.18 An example interaction from the origin to destination E (see map in Figure 6.13) including four decision points. The photo in each case shows the decision point that the user faces at the moment of the next instruction. User button presses are shown in cursive fonts. The photos are a courtesy of Cui Jian.	182

List of Tables

2.1	Summary of approaches towards context-sensitive generation, ordered chronologically with respect to authors, the domain addressed, the context variables involved and the method used.	33
3.1	Annotation scheme for the GIVE corpus grouped by feature types.	62
3.2	Mean values of subjective measure results (adapted from Walker et al. [2000]) based on 156 dialogues (standard deviations of values are given behind the \pm sign.	70
3.3	Mean values of objective measure results based on 156 dialogues, organised in three groups: dialogue efficiency, dialogue quality, and task success. Standard deviations are given behind the \pm sign.	71
3.4	Correlation coefficients between task success and user satisfaction measures (significant at $p < 0.05$), where n.s. means not significant'.	72
3.5	Sample dialogue using the policy optimised jointly. In terms of content selection, the system starts using a high-level navigation strategy, and then switches to low-level as the user gets confused. When the target route segment is navigated, it switches back to high-level. In terms of utterance planning, the system prefers composite presentations for less than three instructions, and incremental displays otherwise. Surface realisation generates only grammatical surface forms without information from a language model.	76
3.6	Sample dialogue using the policy optimised in isolation. In terms of content selection and utterance planning, the agent prefers efficient instruction giving. It uses high-level navigation, aggregates utterances and presents them jointly. In terms of repair, only the immediately preceding utterance is repaired. Surface realisation again generates grammatical surface forms.	77

LIST OF TABLES

5.1	Three examples of (self-)alignment in the GIVE corpus. In the first example, the instruction giver uses the phrase ‘ <i>you want</i> ’ with high frequency and across instructions. In the second example, the instruction giver has a preference for using path instructions including the verb ‘ <i>go</i> ’. In the third example, the instruction giver uses exclusively the verbs ‘ <i>click</i> ’ and ‘ <i>hit</i> ’ in their referring expressions. The number of intervening instructions are shown in parentheses behind each instruction.	110
5.2	Comparison of surface forms generated with graphical models and CFGs against human-authored instructions in terms of Precision-Recall (the higher the better) and KL-Divergence (the lower the better). The first row compares two human data sets and the remaining rows compare both together (Real1 + Real2) against a data set generated with BNs, HMMs, PCFGs and CFGs.	135
5.3	Updated sample dialogue (from Chapter 3, Table 3.5) using the policy optimised jointly including Bayesian Networks for surface realisation. In comparison to the surface realisation choices of the earlier dialogue, the Bayesian Networks here contribute to more natural and human-like language. The content selection and utterance planning strategies are largely unchanged.	137
6.1	Objective metrics for the GIVE evaluation that were logged during interactions. These metrics are very similar to those used to induce a reward function from data in Section 3.5.2.3.	143
6.2	Subjective metrics for the GIVE evaluation and the questions that were asked in order to obtain them. These metrics are very similar to those we used to induce a reward function from data in Section 3.5.2.3.	144
6.3	151
6.4	154
6.5	157
6.6	163

LIST OF TABLES

6.7	Objective and subjective metrics for our systems (J=Joint and I=Isolated) compared with the best systems of the GIVE-2 challenge (NA and S) and the GIVE-2.5 challenge (P1, P2, L, C, CL). * indicates measures taken from Benotti and Denis [2011b] rather than the GIVE challenge.	166
6.8	Objective metrics for the real navigation evaluation. We use category O1 to measure the time that an interaction took and measures O2, O3 and O4 to measure the success and smoothness of interactions.	173
6.9	Subjective metrics for the real navigation evaluation and the questions that were asked in order to obtain them.	173
6.10	179

LIST OF TABLES

Chapter 1

Introduction

Natural Language Generation (NLG) is the task of a machine or computer to map a semantic form to a representation in natural language. It is part of the field of Natural Language Processing and is often seen as the opposite of Natural Language Understanding (NLU). In NLU, the task of the machine is to find a semantic form representing the meaning of an incoming string of words. In NLG, in contrast, the machine’s task is to find the best natural language realisation for an underlying concept.

Stand-alone NLG systems are often used for text generation, e.g. to summarise the weather forecast or produce a patient leaflet. Several detailed example applications are discussed in [Reiter and Dale \[2000\]](#). As part of interactive systems, NLG components are concerned with the generation of output to be presented to a user. An important feature of interactive NLG systems is therefore that the system is able to observe user reactions and take them into account for its own output generation. This thesis will focus on interactive NLG, which allows adaptation to dynamic changes in the system’s environment, including features of the situation and the user. We will particularly focus on instruction generation within an interactive setting, rather than text or paragraph generation.

1.1 Context-Sensitive Situated NLG

Natural Language Generation systems across domains typically face an uncertainty with respect to the best utterance to generate in a given context. The reason is that utterances can have different effects depending on the physical environment, pragmatic circumstances, addressee and interaction history that characterise the context in which they occur. [Malinowski \[1923\]](#) describes the interrelationship between an utterance and its context, *the context of situation*, as follows:

1. INTRODUCTION

'[U]tterance and situation are bound up inextricably with each other and the context of situation is indispensable for the understanding of the words. Exactly as in the reality of spoken or written language, a word without linguistic context is a mere figment and stands for nothing by itself, so in the reality of a spoken living tongue, the utterance has no meaning except in the context of situation' Malinowski [1923], p. 307, cited in Hasan [1985]

This early view of the role of language in context is still as relevant today as it was nearly a century ago. What is even more interesting, it is highly relevant for the development and design of modern NLG systems that interact with human users. Humans can often anticipate the effects that their utterances will have in their given context with high accuracy. Systems, on the other hand, that attempt to make similar predictions can find it a daunting task.

Apart from the context of situation, the linguistic context in which an utterance occurs can affect its form. The psycholinguistic theory of *interactive alignment* states that human interlocutors align their mental and linguistic representations during an interaction through priming.¹ This coordination of linguistic knowledge often leads to more effective and successful interaction when the interlocutors are well aligned than when they are not. Such alignment occurs at all linguistic levels—semantic, lexical, syntactic and phonological—and even percolates between them [Pickering and Garrod, 2004]. The research goal of this thesis is to develop a model of context and adaptivity that generalises to different types of contexts and covers the notions of both the context of situation and of alignment. From a system engineering perspective, we therefore face three challenges to begin with.

- What are the distinguishing features of the situation that make one utterance more felicitous than another?
- How can we formalise the space of choices of the system in a principled way and control the application and execution of choices?
- How can we generalise to unseen circumstances and still be confident that the system will generate reasonable utterances?

¹This means, they adapt to their surroundings by imitating behaviour they observe from an interlocutor, e.g., in an interaction.

1. Two Architectures for NLG

The first question concerns the knowledge of the system and will need to be answered empirically based on human examples. The second question addresses the system’s behavioural potential and the way in which actions are triggered. The third question addresses the issue of generalisability of a behaviour across scenarios. To approach these questions, this thesis suggests using machine learning techniques, in particular hierarchical reinforcement learning and graphical models. A hierarchical reinforcement learner divides an NLG task into a hierarchy of subtasks and learns the best behaviour for the hierarchy by trial and error using a divide-and-conquer approach. The graphical models, particularly Hidden Markov Models and Bayesian Networks, provide feedback to the agent’s learning process in terms of human preferences for surface realisation variants. Both models are trained from human data.

1.2 Two Architectures for NLG

While the specific generation subtasks chosen for a system depend on the system designer, the core tasks that any NLG system needs to address are typically the choice of semantic concepts, their organisation into distinct sub-messages and their realisation as a string of words (see Reiter [1994] and Reiter and Dale [2000] for a detailed overview of generation tasks). Reiter [1994] suggests NLG stages of content determination, sentence planning and surface realisation. The first stage is responsible for producing a first semantic form to be generated. The second stage is responsible for organising this semantic form into a number of clauses and sentences, choose referring expressions and grammatical relationships between messages. The third stage is responsible for producing a surface form.

While related in spirit, this thesis will propose a slightly modified architecture that is suitable for situated NLG. **Situated NLG** can be defined as generation in an enriched physical context, including features of a (real or virtual) environment, such as landmarks and users. Such scenarios can also be referred to as situated domains. The context in this setting is typically not static but undergoes dynamic changes triggered by linguistic or non-linguistic actions by the system or the user. Often, as in this thesis, situated NLG also deals with an additional element of interactivity in that the user can immediately react to the system’s instructions

1. INTRODUCTION

through linguistic or non-linguistic actions. To account for these dynamics, the proposed architecture focuses on utterance generation with constant (non-verbal) feedback from a user. It distinguishes the tasks of *content selection*, *utterance planning* and *surface realisation*. The content selection stage chooses *what to say* in an utterance. It constructs a first semantic form including the level of detail and passes it on to utterance planning. In contrast to Reiter [1994], we also include referring expression generation into content selection, so that utterance planning is exclusively responsible for message organisation and aggregation. Once this was done, the (augmented) semantic form is passed on to surface realisation where a string of words is generated and presented to the user.

Traditionally, NLG systems have used a pipeline architecture to organise the generation process into distinct stages of decision making, each of them addressing a single subtask. Such an architecture for our domain is shown in Figure 1.1. Decisions at different stages are not entirely independent of each other since the information of one module is fed into the next module. Nevertheless, the sequence of decision making is fixed so that decisions at later stages cannot affect decisions made at earlier stages. They are restricted to a local context. Sequential architectures have long been a standard in NLG system design due to their ease of use and modularity. Reiter [1994] also argues that the modular design of the pipeline may resemble human language production, but most importantly, it is efficient and easy to debug from a system engineering perspective. On the other hand, pipeline architectures have been criticised at several occasions, mainly because they do not take the interdependencies into account that exist among different NLG tasks. Several authors have therefore investigated the possibility of treating two or several subtasks jointly [Angeli et al., 2010; Bontcheva and Wilks, 2001; Marciniak and Strube, 2004; Meteer, 1991; Walker et al., 2007]. A discussion is given in Chapter 2.

This thesis will revisit the idea of sharing knowledge among NLG tasks and suggest a joint optimisation model which is able to consider NLG choices at different stages interdependently while still following a modular architecture for scalability and easy maintenance. We can define a **joint optimisation** here as a computational model in which generation subtasks share a set of predefined knowledge variables with other subtasks. For example, the content selection

1. Two Architectures for NLG

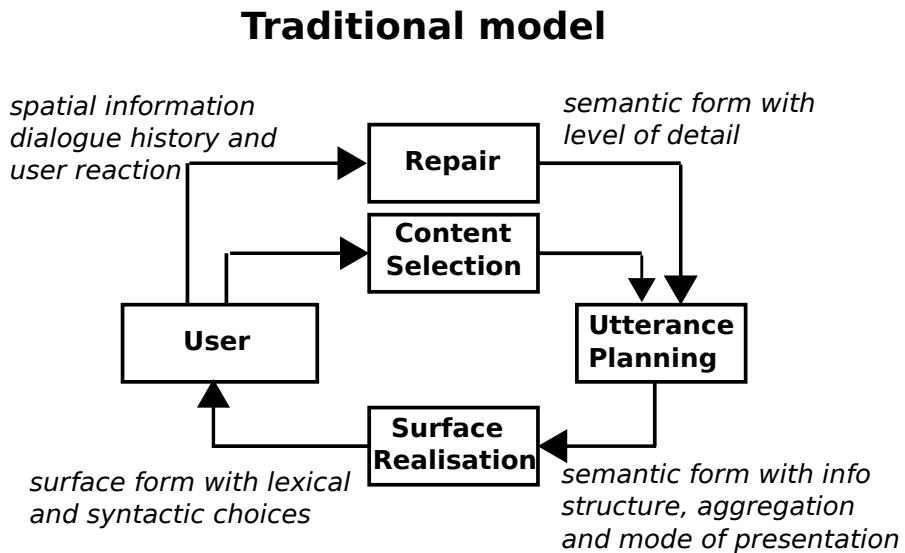


Figure 1.1: A traditional pipeline architecture for NLG systems using sequential decision making for the core tasks of content selection, utterance planning and surface realisation (adapted roughly from [Reiter and Dale, 1997, 2000]).

module can share information on its chosen navigation strategy with utterance planning so that it can be considered for distributing content over messages. In the isolated case, no shared knowledge is available so that each module acts on its own and is blind to the decisions made at subsequent stages. Decision making in the isolated model is sequential and resembles the pipeline architecture for generation. Our joint model *learns* the best sequence of decisions through a trial and error process. By trying alternative sequences of decisions and observing the user's reactions, the system becomes able to predict their consequences under a range of circumstances and execute the most favourable strategy. Figure 1.2 shows the joint architecture we are proposing for situated NLG. Here, the tasks of content selection, utterance planning and surface realisation jointly inform generation by sharing certain knowledge among them so that decisions are made under consideration of trade-offs at different levels and modules.

Some motivating examples can help to see why a joint optimisation is useful. Consider the three scenes with their alternative instructions in Figure 1.3. Some of the given instructions are more felicitous in their given context than others. In

1. INTRODUCTION

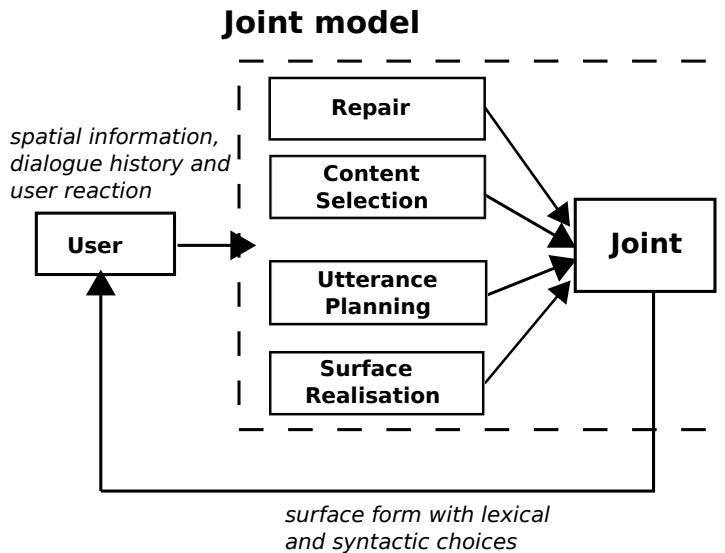


Figure 1.2: An architecture for NLG systems that treats the core tasks of content selection, utterance planning and surface realisation in a joint module that is subject to optimisation. The framework developed in this thesis is based on a joint architecture using hierarchical reinforcement learning.

Scene 1 for example, it seems that both (a) and (b) are equivalent alternatives to refer to the yellow button shown in the picture. Instruction (c) however seems suboptimal. This suboptimality is caused by a lack of communication between the content selection and surface realisation modules.¹ While the surface realisation module knew that in some cases ‘it’ can be used instead of a full noun (‘*the button*’ in this case), it seems to have incomplete knowledge about when exactly this is possible. Such cases can be prevented by sharing information between content selection and surface realisation and jointly optimising them. In this case, the system should have learnt about the trade-off of using less detail in referring to objects that are already part of the interaction history and using more detail throughout the instruction. In the former case, it could then have generated two consecutive instructions *Go to the button by the window* and *Click it*. In the latter case, it could have generated one more detailed instruction *Click the button by the window*.

¹This is true under the assumption that the pronoun *it* is treated by the surface realisation module, not by content selection.

1. Two Architectures for NLG

- 1)  (a) Go and push the yellow button.
(b) Press the button by the window.
(c) Click it by the window.
-
- 2)  (a) Press the green button.
(b) Go to the green button and click it.
 Avoid the red tile.
(c) Go around the red alarm tile.
-
- 3)  (a) Press green.
(b) Go to the lamp by the bed and face it.
 Turn around and go to the green button
 to the right of the cupboard. Push it.
(c) Head to the green button beside
 the cupboard.

Figure 1.3: Example scenes with alternative instructions (some more felicitous than others in their context) from the GIVE environment [Byron et al., 2009].

In Scene 2, we see examples of different levels of granularity chosen by the content selection module. Instructions (a) and (c) both give some information about what to do next but are guided predominantly by efficiency considerations. They leave out important detail. Instruction (b) has found the right balance between efficiency considerations and the user's information need. In instruction (a), for example, the user is likely to step on the red alarm tile that is visible in the foreground (because they did not receive a warning not to do so), trigger an alarm and fail the navigation task immediately. In instruction (c), a second instruction would need to follow to complete the referring expression that could

1. INTRODUCTION

then successfully guide the user to press the button without activating the alarm. While this example does not necessarily require a joint optimisation of subtasks, it illustrates a potential application of reinforcement learning with a divide-and-conquer approach to balance different trade-offs, here efficiency vs detail, over time and learn an optimal policy.

Scene 3 again shows examples of three instructions that differ in their level of granularity. Instruction (a) is likely to not be successful since it does not uniquely identify the referent button. Instruction (b) gives a unique reference but is very verbose. Only instruction (c) seems to balance the trade-offs between the user’s information need and cognitive load. This example illustrates the need for communication between the content selection and utterance planning modules during generation. While instruction (b) should allow the user to find and push the intended referent, the instruction contains too much unnecessary detail so that a user may find it hard to memorise. Since the system here was predominantly guided by providing complete information, it would have been advantageous to spread the information over a sequence of simpler instructions to ease the user’s cognitive load. The system could for example have generated three separate instructions *Go to the lamp by the bed and face it*, *Turn around and go to the green button to the right of the cupboard* and *Push it*. The consideration of trade-offs such as information need and cognitive load becomes possible through a joint optimisation of content selection and utterance planning.

As demonstrated in some of these examples, the interdependencies among NLG tasks are multiple and a reinforcement learning agent will typically face the trade-offs of efficiency, the users’ information need and their cognitive load. The amount of information included in the content selection module should be taken into account by the utterance planning module so that an appropriate presentation strategy is chosen for the utterance. Surface realisation should make its decision with respect to a language model, but not choose surface forms at the cost of losing information (say if one surface form is more informative than another but less frequent). We will argue for an approach that optimises NLG by taking the user’s knowledge into account. The remainder of this thesis will present a joint optimisation framework for situated NLG based on hierarchical reinforcement learning and graphical models.

1. Outline of the Thesis

1.3 Outline of the Thesis

This thesis makes a contribution to the field of robust,¹ flexible and adaptive NLG for interactive systems. The framework suggested will be based on machine learning techniques with a special focus on hierarchical reinforcement learning.

We will begin with a comprehensive review of context-sensitive approaches to NLG in **Chapter 2**. In particular, we will compare the advantages and limitations of *rule-based* approaches to context-sensitive NLG, *planning and constraint-satisfaction* approaches, *trainable* approaches and several others. We will then discuss the current state-of-the-art in using *reinforcement learning* for NLG, in applying a *joint* treatment of subtasks to NLG or dialogue systems and in using *graphical models* as generation spaces for surface realisation.

We will then give a detailed introduction into *hierarchical reinforcement learning* and its application to situated NLG in **Chapter 3**. Here, we will also design and train a first learning agent for NLG in situated interaction which will be extended and modified in later chapters.

In **Chapter 4**, we will focus on content selection and introduce the notion of a *hierarchical information state*. This can be used to systematically pre-specify prior knowledge or human preferences of a domain. We will use decision trees to discover human preferences from a corpus of situated human-human interactions and use them to constrain the learning agent's action selection process.

Chapter 5 will focus on surface realisation. We will propose the use of graphical models as representing generation spaces of the domain. The graphical models will be integrated into the developed learning framework so as to inform the agent's surface realisation learning process. In addition, we will introduce a method for balancing the proportion of alignment and variation in surface re-

¹By robust in this context, we mean NLG systems that are able to take reasonable actions even in situations which have not been encountered in human training data but differ in certain respects. The agent should then be able to generalise and take the best possible action.

1. INTRODUCTION

alisation based on Gaussian sampling. Chapters 3, 4 and 5 represent the main content chapters of this thesis.

An evaluation of the developed framework will be presented in **Chapter 6**. We will compare a system optimised jointly—using the framework developed in Chapters 3 to 5—against a system optimised in isolation—using hierarchical reinforcement learning but without shared state variables. We will compare both systems based on user satisfaction, task success and a range of objective and subjective metrics. Furthermore, an experimental study will be presented that transfers the developed method to a new, but related domain and tests its generalisability and transferability to new domains.

Finally, **Chapter 7** will draw conclusions and present directions for future research.

Chapter 2

Related Work

2.1 Introduction

The ability of a Natural Language Generation (NLG) system to adapt to different users and circumstances in multiple ways has been one of the core research interests in NLG from the beginning of the field. Adaptation can take many forms ranging from content selection decisions over surface realisation to rhetorical and discourse structure and many others. For content selection, for example, a system can choose different levels of detail depending on the user’s prior knowledge of a domain. Similarly, different word choices can be made for experts and novices of a particular domain or application. Rhetorical relations can help to generate argumentative texts where the arguments themselves can be tailored towards different groups of users. Apart from a wide range of applications, a wide range of methods have been proposed to address the problem of *context-sensitive NLG*. They include *rule-based* approaches to NLG, *constraint satisfaction* and *planning* approaches and *trainable* approaches. While NLG has been one of the last subfields of Natural Language Processing to adopt statistical methods on a larger scale, the past decade has seen an interesting bulk of work on the topic of context-sensitive NLG using machine learning. This chapter will review the most important strands of work in context-sensitive generation while giving a special focus to context-sensitive accounts using machine learning.

Subsequently, we will review accounts of reinforcement learning for NLG. These have generally focused on specifying an abstract goal which a learning agent is set to solve by discovering an optimal strategy automatically through training using a data-driven user simulation and reward function. We will discuss how the framework developed in this thesis relates to previous work on NLG using reinforcement learning.

2. RELATED WORK

In addition, we will review the development of the idea of a joint treatment of interrelated subtasks within a domain. While this idea has existed in the research community for more than two decades, it has been treated statistically only very recently and has not been applied within a reinforcement learning-based optimisation framework as suggested in this thesis.

Finally, we will review previous work on using graphical models for surface realisation. While this thesis shares the general idea of training graphical models based on human data for natural and variable surface realisation, it goes beyond previous work by offering a joint optimisation perspective under which surface realisation choices are considered directly in conjunction with content selection and utterance planning.

2.2 Context-Sensitive Language Generation

As noted above, context-sensitive adaptation to users and situations has been one of the core interests of the NLG community from its early beginnings. This section will discuss some of the most important research in this area giving a special focus to machine learning approaches.

We assume that the concept of *context-sensitive generation* here refers to systems that are able to adapt their linguistic output to dynamic aspects of user or situation. We do not consider systems that generate output based on (static) spatial properties, such as configurations of objects, but do not show any adaptation apart from this as is the case with many referring expression generation algorithms [Dale, 1989; Dale and Haddock, 1991; Dale and Reiter, 1995; Dale and Viethen, 2009; Horacek, 2005]. While these approaches are to some extent sensitive to properties of their spatial context, they are not part of context-sensitive generation as understood in this thesis.

2.2.1 Rule-Based Approaches to Generation

Context-sensitive NLG as an investigation into systems that are able to adapt their output towards a range of different users and contexts has its roots in

2. Context-Sensitive Language Generation

rule-based accounts that treat NLG as a process of *explicit choice*. Under this perspective, generation can be seen as a sequence of choices that ultimately lead deterministically to one single output and can be solved using rules [Mann and Matthiessen, 1983; Oberlander et al., 1998; O'Donnell, 1996]. Several approaches have also explored generation based on unification grammars [Elhadad, 1993; Kay, 1985; McKeown, 1985; Shieber et al., 1990; Wilcock and Matsumoto, 1998], Tree Adjoining grammar [Joshi, 1987; Nicolov et al., 1996; Stone and Doran, 1996] or Systemic Functional Grammar [Bateman, 1997]. Other approaches include Lavoie and Rambow [1997] who generate language based on Meaning-Text Theory [Mel'cuk, 1988], Calder et al. [1989] who suggest generation based on Unification Categorial Grammar and Reiter and Mellish [1992] who use classification. Ward [1994] gives a comprehensive overview of the *generation as choice* paradigm.

For context-sensitive generation in particular, this means that certain variables of the context, the environment or the user can lead deterministically to a generation outcome. The generation process is typically guided by rules. Bateman and Paris [1989] present an early account of user modelling in context-sensitive NLG in which the system can adapt its descriptions of electrical circuits to the user's level of expertise in that domain. Similarly, Paris [1993] generates descriptions of various technical devices tailored towards the user's prior knowledge. Bateman and Teich [1995] develop a system that generates biographies based on the current text genre.

More recently, Paris et al. [2004] have presented Myriad, a general architecture for the development of information retrieval systems that support a dynamic and evolving model of context taking into account features of the user, their domain of application, tasks they commonly perform and devices they typically use. They present a number of example applications that allow users to retrieve personalised results from web searches that are specifically tailored to their individual needs.

White et al. [2010] present the FLIGHTS system which is able to present flight-related information to users that is tailored towards their individual preferences. NLG in the FLIGHTS system is concerned with content and attribute

2. RELATED WORK

selection, selection of referring expressions, information structure and realisation units for speech synthesis. All of these decisions are made in an integrated fashion so as to ideally emphasise the user preferences and arising trade-offs. For a user preferring to travel on the cheapest flight, for example, the system will present the cheapest flight using a prosodic realisation that appropriately puts the price attribute in focus, such as ‘the *cheapest* flight’, where cursive fonts indicate prosodic emphasis. In addition, alternative flight options will be presented that mark the trade-offs the user will have to make by both lexical choice and intonation, such as ‘there is also a *direct* flight, but it is more expensive’. Here, the inclusion of ‘but’ marks a trade-off for the user. White et al. present a human evaluation showing that users significantly prefer the emphatic prosody of the system over its baselines.

Dethlefs et al. [2011] present an approach for context-sensitive generation of outdoor route instructions in urban environments that takes different properties of the user and the environment into account. Particularly, route instructions are specifically tailored towards different degrees of prior knowledge that users have of the navigation environment, that is, their *familiarity* with the area. Moreover, route instructions are sensitive to salient geographical properties in the environment, such as big or frequently travelled streets and landmarks that are salient in the environment due to their function, size, proximity to the road, permanence or other salience properties.

Explicit choice approaches have a distinct advantage over automatic or statistical approaches in that they give complete control to the system designer. This allows the specification of detailed linguistic or psychological prior knowledge so that system behaviour can be directly informed by psycholinguistic studies and experimental evidence, enhancing the cognitive plausibility of behaviour. A disadvantage of knowledge-based approaches, on the other hand, is that system development and maintenance can be expensive and time-consuming. Also, no automatic optimisation is possible that can discover fine-grained rules in a set of human data. As a result, rule-based systems are often not able to adapt to new situations or unseen circumstances on the fly. Their adaptability is thus

2. Context-Sensitive Language Generation

constrained to those use cases for which they were developed and, to a significant degree, dependent on the skills of the system designer.

2.2.2 Generation As Planning

Early approaches to planning have typically defined a set of logical constraints, capturing linguistic or contextual knowledge, and then used planning to find one or several ways to meet these constraints [McDonald, 1976; Meteer et al., 1987]. An early account of context-sensitive planning is Patten [1988], who defines a set of semantic goals, including features of context, and then finds lexical and syntactic realisations that meet these semantic goals. Alternative early planning accounts include Cohen and Perrault [1986] who present a planning account that takes speech acts into account, and Heeman and Hirst [1995] who develop a model for collaborative referring expression production between interlocutors.

Several approaches also exist for using hierarchical planning in NLG. Moore and Paris [1994] present research on generating explanations in a system that is able to reason about its own previous utterances to interpret follow-up questions or clarifications, for example. They use Rhetorical Structure Theory [Mann and Thompson, 1988] to represent hierarchical discourse relations within text. These can then be used as planning operators in text generation. For other approaches towards hierarchical planning, see Hovy [1991] and Appelt [1992].

A more recent approach to constraint satisfaction is Piwek and van Deemter [2007], who present an approach to generating scripted dialogues, such as found e.g. in theater, on television or radio. Their approach addresses particularly the case where several conflicting constraints need to be satisfied together. As an example, they discuss the constraints of the *length* of a dialogue versus its *emphasis* under the assumption that emphasis typically contributes to length. The challenge is then to balance both constraints against each other if it is not possible to satisfy them both optimally.

While earlier work on planning has often relied on ad hoc planners suited to particular domains, more recent work has aimed for a clear separation of plan-

2. RELATED WORK

ning and domain knowledge [Koller and Petrick, 2011]. Stone et al. [2001]’s SPUD system represents an approach to microplanning that is based on generation with LTAG. They explicitly associate the communicative intend of an utterance with its linguistic realisation, so that a generation plan always needs to meet the communicative goal. In this way, different realisation candidates can be evaluated. Incidentally, this approach also combines pragmatic, semantic, syntactic and lexical constraints into a joint decision making process. Koller and Stone [2007] present an approach to sentence generation which works in combination with principled AI planners. In this work, all semantic and pragmatic constraints introduced by a surface form are directly encoded into the planner so that they can be taken into account for choosing the best surface realisation given a concept and context as well as the constraints specified by the realisation itself. Again, sentence planning and realisation are conflated into a single process.

Other recent approaches to planning include Garoufi and Koller [2010], and Garoufi and Koller [2011a]. Garoufi and Koller [2010] extend the approach by Koller and Stone [2007] to situated generation where the goal of their planner is to generate non-ambiguous referring expressions in a virtual 3D environment. This goal can be achieved by exploiting and manipulating the linguistic and non-linguistic generation context. Garoufi and Koller [2011a] extend this model further to deal with alternative surface realisations where they aim to find the one that a user will find easiest to resolve. This approach is based on a maximum entropy model.¹

Planning approaches have been shown to be very successful for context-sensitive NLG. They reliably meet their specified goal and typically have several ways of doing so, in case one of them fails. Research on reinforcement learning for NLG is in several ways related to treating NLG as a planning problem. It can be seen as a possible solution to statistical planning in which well-studied algorithms are used for finding action strategies for NLG tasks. In contrast to other approaches,

¹As such, it is a hybrid model that lies between the planning approaches discussed in this section and the trainable approaches discussed in the next section. We decided to list it here due to the core of the authors’ model being planning.

2. Context-Sensitive Language Generation

RL is particularly suited for tasks in which we are unsure of the best strategy to achieve a goal and wish the system to find an optimal policy automatically from interactions with the environment and/or a user.

2.2.3 Trainable Generation

The idea of trainable generation was first introduced as an *over-generation and ranking* task in seminal work by [Langkilde and Knight \[1998\]](#). They suggest a statistical approach to generation which, in a first step, generates a set of possible lexical-syntactic renderings of an underspecified semantic input, and in a second step, ranks these according to corpus-based n-gram probabilities. A similar direction is pursued by [Oh and Rudnicky \[2000\]](#) who use corpus-based n-grams models for attribute selection for content planning, and over-generation and ranking for surface realisation. Their approach is based on spoken corpora and specifically tailored towards spoken dialogue systems. Several authors followed the basic intuition behind over-generation and ranking in following years. [Bangalore and Rambow \[2000a\]](#) extend Langkilde and Knight’s model by using tree-based representations of syntactic structure and lexical choice [[Bangalore and Rambow, 2000b](#)]. [Ratnaparkhi \[2000\]](#) suggests to rank alternative surface realisation using a maximum entropy model. While these approaches can be said to mark the beginning of trainable generation systems, none of them focused on context-sensitive generation, in particular. We will turn to such approaches in the following.

[Stoia et al. \[2006\]](#) present an algorithm for generating noun phrases for a spoken interaction task in a virtual 3D environment. Their algorithm is based on the output of a decision tree classifier using features of the environment such as the user’s viewing angle, their distance from the target noun phrase referent, number of its distractors, visibility and discourse history. They present results from a human rating study in which noun phrases generated with their algorithm were rated equivalently or better than their original human counterpart in 62.6% of cases.

2. RELATED WORK

Foster and Oberlander [2006] generate facial displays for a talking head that accompanies spoken output of a system. The facial displays are based on a human corpus and are sensitive to the utterance context in terms of its words, its pitch-accent specification, and its domain context. They are also sensitive to whether or not the property has been presented before, stands in contrast to another property and the expected user evaluation of the presented property. Their results show that while users prefer generation models for facial displays that take the context into account, they also showed a significant preference for models that produced variation in their output.

Demberg and Moore [2006] present an algorithm that combines traditional user modelling with automated clustering for information presentation of flight information. They use a cluster-based tree structure to decide which information to present to the user, based on their preferences, in case there are several possibly relevant options to present. The tree structure guides the dialogue flow and the presentation. The overall trade-off they aim to address is to give users the feeling of having gained a good overview of their choices, while always just presenting options that the user is likely to be interested in. They present results showing a significant increase in user satisfaction over a baseline.

Walker et al. [2007] present the SPARKY system which is able to adapt its surface realisation to individual users' preferences by generating a large number of possible realisations in a first realisation stage, and then ranking them using boosting based on a user-specific corpus or user feedback in a second step. They present results showing that sentences generated by SPARKY receive significantly better human ratings than a random sentence realiser and are rated on average 10% worse than the best rated human sentences. The authors further show that generation models trained (and tested) on data from individual users achieves significantly better performance than training from the data of an average population of users. The context here can therefore be said here to be constituted by an individual user and their surface realisation preferences.

2. Context-Sensitive Language Generation

Roth and Frank [2010] present a minimally supervised learning approach to the generation of route instructions for outdoor environments. They use the Expectation-Maximising (EM) algorithm to align geographical route representations with corresponding linguistic realisations that were taken from an annotated corpus of human data. They argue that route instruction generation can be a highly context-dependent task when adapting to specific spatial configurations and environments also including diverse spatial objects that may be present.

Training from unannotated data, Chen et al. [2010] present a system that learns to interpret and generate language based on pairs of action sequences and textual descriptions of RoboCup games. A particular challenge is that the action sequences are ambiguous in that not every action is described in the corresponding text. The authors' best performing system in terms of surface realisation was optimised for precision by comparing generated system output against human-authored text. For content selection, the authors train their generator using a variant of the EM algorithm to estimate which events are worth including in a textual description and which are not. For a related application in language understanding, Vogel and Jurafsky [2010] use apprenticeship learning to map (ambiguous) route instructions onto executable spatial actions.

Trainable approaches to generation are typically based on human corpus data that is domain- or task specific. This often allows the mimicking or modeling of behaviour of a majority of human subjects. Data-driven approaches therefore do not entirely rely on the intuitions of individual system designers. In addition, approaches such as those using supervised learning are often faster to develop or transfer to new domains than rule-based or planning approaches that need to be designed for particular domains. Supervised learning approaches are discussed more in Chapter 4. While often classified as separate learning paradigms, supervised learning and reinforcement learning are related in that both are concerned with training an agent to adopt a certain action policy. Supervised learning approaches tend to be instructive in that the agent learns from labelled examples which directly specify the correct action to take. Reinforcement learning approaches, on the other hand, are more evaluative in that they provide a reward

2. RELATED WORK

and let the agent discover the best strategy to maximise long-term rewards automatically. This makes it well suited for the optimisation of sequential decision making problems such as situated interaction, in which an NLG system needs to generate an effective and coherent sequence of instructions to a user.

2.2.4 Other Approaches

Other approaches to context-sensitive generation have explored models based on game-theory, based on permissions, obligations and prohibitions and based on salience measures. They are summarised in the following.

[Golland et al. \[2010\]](#) treat language generation as a game-theoretic model in which a set of semantic and pragmatic constraints need to be satisfied. They address the task of generating spatial descriptions so as to allow the hearer to identify a target object as easily as possible. They present results showing that a generation model that takes the pragmatic effects of its utterances on the hearer into account significantly outperforms a model that does not.

[Bouttaz et al. \[2011\]](#) generate text descriptions of online resources, where the context for generation is defined by permissions, prohibitions and obligations concerning particular users or resources. In their domain of generating within an internet platform for business or research purposes, which users can use to share resources or messages, different users of the platform may have different permissions to access information stored on the platform which need to be taken into account by the NLG system.

[Chiarcos \[2011\]](#) presents an approach for context-sensitive referring expression generation that is based on a theoretical model of salience grounded in cognitive and functional linguistics. This model takes two measures of salience into account, one focusing on hearer-related salience and one focusing on speaker-related salience. Given a set of candidate realisations for a referring expression, the best candidate can be chosen according to both salience measures.

2.3 Reinforcement Learning for NLG

Reinforcement learning agents optimise sequential decision making problems by mapping situations to actions and maximising a long-term reward signal. The main ingredients of a reinforcement learning agent are therefore

- a set of states S representing the agent's knowledge of the situation and user,
- a set of actions A representing the agent's behavioural potential,
- a stochastic state transition function T which specified the next state s' of the agent for taking action a in state s , and
- an objective reward function R that assigns a numeric value to the agent for taking action a in state s .

Please see [Sutton and Barto \[1998\]](#); [Szepesvari. \[2010\]](#) or Chapter 3 for a comprehensive overview of the reinforcement learning model.

Reinforcement learning has been a popular method applied to dialogue management for about a decade [[Henderson et al., 2008](#); [Levin et al., 2000](#); [Pietquin and Dutoit, 2006](#); [Schatzmann and Young, 2009](#); [Singh et al., 2002](#); [Walker, 2000](#)], where it has been appreciated especially for its ability of automatic optimisation, discovery of fine-grained behaviour from human data and adaptability under unseen and uncertain circumstances [[Williams and Young, 2007](#)]. For NLG, [van Deemter \[2009\]](#) presented an informal argument for viewing NLG as a choice process in which several trade-offs have to be considered and balanced. Reinforcement learning has been adopted by the NLG community especially with a focus on optimising generation for interactive systems by [Janarthanam \[2011\]](#); [Lemon \[2008\]](#); [Rieser and Lemon \[2011\]](#).

[Janarthanam and Lemon \[2010a\]](#) present a system for referring expression generation within troubleshooting dialogues, where the system is able to tailor its generation choices towards the user's individual prior knowledge of the domain. The dialogue domain addressed is a setting where a system assists a user in setting up a broadband connection so that several choices for referring to technical objects

2. RELATED WORK

arise. They range from using technical *jargon*, such as saying the ‘broadband filter’, to using *descriptions*, such as ‘the small white box’. An interesting feature here is that users’ knowledge will typically change during the interaction due to a learning effect and the system then needs to adapt to a constantly changing user model. The crucial trade-off for the system is therefore to adapt maximally to the user’s individual information need: the system should not use terms that the user does not understand, but should also not engage in unnecessarily lengthy descriptions.

Training of the learning agent was done using the SHARSA algorithm [Shapiro and Langley, 2002] against a user simulation that was estimated from data collected in a Wizard-of-Oz setting [Janarthanam and Lemon, 2009a]. Here, user responses are simulated based on the system dialogue act, the user’s prior knowledge of the domain and previous clarification requests given the referring expression under consideration [Janarthanam and Lemon, 2009b].

The overall generation model was evaluated against a number of baselines which used less individualised adaptation (such as switching to descriptive mode whenever the user requests clarification or switching to a jargon mode whenever the user complains about too much information) in [Janarthanam and Lemon, 2010b]. Results showed that the learnt policy produced significantly more accurate adaptations than the baselines, led to shorter dialogues and higher task completions. Users also indicated that they found it easier to identify referent objects using the learnt policy than using a baseline.

Rieser et al. [2010] present an optimisation approach for Information Presentation in a tourist information domain, specifically with a focus on restaurant recommendations. The system is faced with two main choices: (1) which information presentation strategy to choose among *summary*, *comparison*, *recommendation* and ordered combinations of these, and (2) which attributes to include in the presentation of *cuisine type*, *food quality*, *location*, *price range* and *service quality*. The main trade-off the learning agent needs to optimise is to present enough information to users to give them the feeling of having a good overview of the choices, but at the same time keep utterances short and understandable.

2. Reinforcement Learning for NLG

The learning agent was trained using a reward function that optimises the number of database hits presented (in relation to all hits), the sentence length and the user reaction. For the last, the authors hand-coded numeric values associated with each possible reaction of *user selects an item*, *user adds further constraints to the search* and *other*. The reward function¹ as well as the n-gram-based simulated environment were estimated from data collected in a Wizard-of-Oz setting [Liu et al., 2009]. Optimisation was performed using the SHARSA algorithm. In a simulation-based evaluation, Rieser et al. [2010] compare their learnt policy with a supervised learning baseline that was induced using the rule learner RIPPER [Cohen, 1996]. Interestingly, results show that the former significantly outperforms the latter. While the RL-based policy attains up to 91.5% of the maximally possible rewards, the supervised learning-based system only attains up to 87.6%.

In addition to the simulation-based results, Rieser et al. [2011] present a human evaluation study of the optimised information presentation strategies for tourist information. They compare their system against a baseline that relies exclusively on recommendations. Results show that the learnt policy outperforms the baseline in terms of objective task success. Users were more successful using the optimised system. For the subjective metrics, however, as well as subjective task success, users expressed a slight preference for the baseline system. A possible explanation offered by the authors is that the learnt system achieves a higher *effectiveness* —perhaps to some extent at the cost of *efficiency* due to their system presenting information in a more lengthy fashion than the baseline.

Janarthanam et al. [2011] use reinforcement learning to optimise the generation of temporal expressions. They train a learning agent in a simulated environment based on a hand-crafted reward function that assesses the *user preference* and *length of the expression* as well as the likelihood of the user requesting *clarification*. In a human evaluation study, they show that users interacting with the learnt policy have a significantly higher subjective task success and user satisfaction than users interacting with any of the baselines. There was no significant

¹A previous study that used an alternative reward function based on the number of attributes to present to the user is presented in [Rieser and Lemon, 2009].

2. RELATED WORK

difference in objective task success, however.

Reinforcement learning has also been applied to other natural language processing tasks. [Branavan et al. \[2009\]](#) use reinforcement learning to map instructions in natural language, such as user manuals, to executable action sequences. As in other RL tasks, the quality of their learnt action sequences is determined by the reward function during training. Since their reward function is based on task completion, they do not require annotations, which is an advantage. In contrast, RL applications in dialogue or generation typically need to be trained in interaction with human users, which makes training more expensive. Even though simulated environments can be used, they often rely on linguistic or pragmatic features which may require annotation, depending on the domain. One possible solution to mitigate this problem has been to use Wizard-of-Oz data collections [Rieser and Lemon \[2008\]](#), which automatically log wizard actions and therefore can be used to bootstrap simulated environments from small data sets.

A central characteristic of RL-based approaches is that they tend of specify abstract system goals, such as ‘help the user set up the broadband connection without using words they don’t understand and without unnecessary descriptions’, or ‘help the user find a restaurant they like without presenting every possible option to them, but still give them a good overview of the choices’. The system is always just told *what* to achieve, but never *how* to achieve it. It is then the learning agent’s objective to try different strategies of achieving its goal and discovering the best. RL agents are able to learn robust and flexible action policies due to the high number of training episodes they experience, which allow them to explore all possible policies and finally converge to the optimal policy. A particularly high degree of adaptability is achieved by learning agents with the capability of assessing and re-learning their behaviour during interactions so as to adapt to individual users and contextual properties. While these are important advantages in practice, they come at a cost: learning can only be successful given a sensible reward function, typically estimated from human data, and a variable and extensive user simulation, which is also usually based on a data collection.

2. Joint Treatment of Subtasks

Unfortunately, previous approaches using reinforcement learning for NLG have relied on flat reinforcement learning which is typically restricted to small-scale domains due the *curse of dimensionality*, the problem that the learning agent’s state representation grows exponentially according to the number of variables taken into account. This makes the flat reinforcement learning setting infeasible for large, or real-world, systems. In addition, previous work has relied exclusively on template-based generation which does not take advances in statistical or trainable surface realisation into account. Finally, subtasks have not been treated jointly, despite their ubiquitous interrelationships. This thesis will demonstrate how these limitations can be overcome by using a joint learning architecture that is based on hierarchical reinforcement learning with graphical models. The hierarchical framework will help us to develop more complex systems that are applicable to real-world scenarios by using a divide-and-conquer approach. The graphical models will support variable and corpus-based surface realisation that is jointly optimised with content selection and utterance planning choices.

2.4 Joint Treatment of Subtasks

A number of recent studies have presented evidence in favour of a joint treatment of tasks. Joint treatments have been shown to outperform their isolated counterparts in domains containing interrelated decision making. While the idea of a joint optimisation, using machine learning, is relatively new, the underlying philosophy of treating interrelated subtasks jointly is not. It can to some extent be seen as a reincarnation of the *generation gap* problem raised by Meteer [1991], who has argued for an increased awareness of earlier modules in the generation process of their effects on later modules. For the machinery discussed in this section, please see Mitchell [1997] for machine learning techniques and Jurafsky and Martin [2009] for others, such as computational grammars or weighted finite state machines.

Stone and Webber [1998] present an early account of using a joint representation of syntax and semantics for generating language so as to maximise the

2. RELATED WORK

textual economy, the degree to which the current sentence is efficient by allowing the user to make inferences with respect to the rest of the discourse. They use Lexicalised Tree-Adjoining Grammar (LTAG) to incrementally build up grammatical structures, where each elementary LTAG tree is associated with a logical form indicating the tree’s contribution to the semantics and pragmatics of the discourse.

Similarly, [Marciniak and Strube \[2004\]](#) present an approach to route direction generation based on Lexicalised TAG with discourse features and instance-based classification. Since the classifiers in their approach directly determine grammatical structures, which are then represented as LTAG structures, traditional NLG stages are also treated in a unified, rather than isolated, fashion. The classifiers were trained from human corpus data and use semantic and discourse features to choose the best grammatical structure based on a route giving situation. [Marciniak and Strube \[2005\]](#) extend this model into a more general argument in favour of a joint optimisation in NLP. Their main argument is that if a task can use the output of a preceding task to improve its performance, then in many cases the reverse is also true. Therefore, the sequential model used in the pipeline is inappropriate and can deteriorate system performance. The authors present evidence in terms of a classification-based NLG system that uses integer linear programming for route instruction generation. Their optimisation framework then aims to minimise the costs of label assignments by considering the global cost involved in generating an instruction, rather than the costs of local decisions. In this way, generated utterances become globally optimal across eight different grammar-related tasks. The authors show that these utterances are superior to those generated using sequential architectures.

[Bulyko and Ostendorf \[2002\]](#) present an approach for jointly optimising NLG and speech synthesis based on weighted finite-state transducers. Instead of passing a single word sequence to the synthesiser for realisation, they pass an annotated network of choices which is then considered jointly with the choices for prosody and speech units given the costs associated with different combinations. In this way, they are able to synthesise the utterance that sounds most natural

2. Joint Treatment of Subtasks

given all involved trade-offs. In a human rating study, results showed that their system was perceived better than the baselines more than two thirds of the time.

Nakatsu and White [2006] follow Bulyko and Ostendorf [2002] in treating NLG and synthesis jointly: they use a discriminative re-ranker to select surface realisations that are predicted to sound natural when synthesised. They train Support Vector Machines on a set of generator and synthesiser features mapped with human judgements of the naturalness and quality of their synthesised output. They present results from a cross-validation study that indicate an improvement of roughly two thirds over the baseline.

Stone et al. [2004] use a database of collected speech and gestures to develop an Embodied Conversational Agent that is able to generate synchronised speech and gestures. They define a cost function that allows the agent to identify sequences of speech and motion that combine in a meaningful way. In this way, they are able to show how speech and motion support each other and reach their peak of emphasis simultaneously.

Angeli et al. [2010] present a robust and domain-independent generation system that employs a joint treatment of content selection and surface realisation. In their approach, each generation decision is handled by a log-linear classifier that has access to all other previous decisions, i.e. the full generation context. They evaluate their approach on the generation of sportscasting for RoboCup games and technical and common weather forecasts and show that a joint treatment achieves better accuracy and human ratings than an isolated treatment which uses only local features.

Lemon [2011] presents a joint optimisation approach to NLG, particularly information presentation, and dialogue management in the area of information presentation for restaurant recommendations. He shows that using reinforcement learning for the optimisation, the jointly optimised policy can learn when it is most advantageous to present information to the user or when to ask for more details to refine the query. The policy optimised in isolation made less fine-grained

2. RELATED WORK

choices especially with respect to the number of items that need to be presented.

[Cuayáhuitl and Dethlefs \[2011a\]](#) present a reinforcement learning approach to spatially-aware dialogue management by optimising it jointly with route planning in a wayfinding domain. They show that the spatially-aware system—optimised jointly—generates the shortest possible route by adapting to individual users' prior knowledge, guiding them past landmarks they are already familiar with and avoiding junctions where they are likely to get confused. The system that was optimised in isolation adapted only to generic information such as guiding users past landmarks that are generally well known. In addition, while the isolated system provided route instructions from origin to destination, the joint system learnt to ask users for knowledge of intermediate landmarks to give more efficient route instructions.

Most recently, [Zarriess and Kuhn \[2013\]](#) develop a model that jointly treats discourse-level referring expression generation and surface realisation. The authors try different orders of generation by alternating the sequence in which modules are called. As other work has also shown, the results confirm that allowing joint knowledge variables or, in this case, allowing revisions, leads to better performance than using a standard pipeline architecture.

We suggest a generation task that integrates discourse-level referring expression generation and sentence-level surface realization. We present a data set of German articles annotated with deep syntax and referents, including some types of implicit referents. Our experiments compare several architectures varying the order of a set of trainable modules. The results suggest that a revision-based pipeline, with intermediate linearization, significantly outperforms standard pipelines or a parallel architecture.

All of the approaches discussed above acknowledge in one form or another the advantage of a joint form of decision making. An early argument in favour of local decision making in the form of the pipeline architecture was presented by [Reiter \[1994\]](#). He observes that despite being based on different linguistic theories, NLG systems in the early nineties had converged onto a consensus architecture which

2. Joint Treatment of Subtasks

follows a pipeline process dealing first with tasks of content selection, followed by sentence planning and finally surface realisation. Reiter speculates that the main reasons for this common architecture are to be found in psycholinguistic findings on the one hand and in engineering principles on the other. For example, there is abundant evidence for modularised human language production facilities in that brain damage often affects some parts of human language production, but not others. He further argues that a modularised architecture is desirable from an engineering perspective because it is efficient and easy to debug.

Many of the joint optimisation architectures discussed above (e.g., [Angeli et al. \[2010\]](#), [Lemon \[2011\]](#) and [Cuayáhuitl and Dethlefs \[2011a\]](#)) work essentially by making additional knowledge available to the components involved. Typically, this is knowledge that has traditionally been specific to one module of the system and is now shared between two or more modules in order to achieve a joint knowledge base on which to base decisions. These joint architectures have deliberately not attempted to share their full knowledge base, which would be computationally expensive as [Reiter \[1994\]](#) observes. Instead, they have shared small parts of knowledge which the system designer expected to positively affect performance. In this way, they are computationally scalable and do not sacrifice the benefits of a modularised architecture.

Based on similar considerations to allow more communication between generation modules and maintain the advantages of the pipeline is work presented by [Bontcheva and Wilks \[2001\]](#). The authors deal with a text generation task in which user preferences for textual content sometimes only become apparent at the surface realisation stage (because they depend on the length of generated texts). At this point it may therefore become necessary to backtrack in order to modify content selection choices. The authors solve this problem by suggesting a recursive pipeline architecture which allows feedback loops from surface realisation to content selection.

A further approach to considering NLG decisions jointly without sacrificing the pipeline are systems like SPaRKy [[Walker et al., 2007](#)]. In such systems, sentence generation takes an overgeneration and ranking approach in which a randomised set of alternative sentence plans are generated in a first step, which are then ranked in a second step according to a boosting score that predicts user

2. RELATED WORK

ratings of the outputs. Finally, a surface realiser chooses the best form from the ranked list. In this approach, joint decision making is possible in that an n -best list of alternatives is passed between modules, which can then each be considered in the next module.

This thesis extends previous work by developing a context-sensitive generation framework that automatically optimises system utterances while assessing the joint effect that content selection, utterance planning and surface realisation decisions have on user reactions to the utterance within a situated generation context.

2.5 Graphical Models for Natural Language Generation

A graphical model can be defined as a visualisation of the structure of a probabilistic model. It consists of nodes, representing random variables of interest, which are connected by edges that express probabilistic relationships between the variables. Graphical models allow the inspection of properties of a probabilistic model from the graph that is used as a representation. Please see [Bishop \[2006\]](#), Chapter 8, for details.

Related work on using graphical models for surface realisation has investigated the use of *lattices*, *probabilistic context-free grammars (PCFGs)*¹ and *Dynamic Bayesian Networks*. The intuition behind all of these approaches is to limit the development effort involved in hand-crafting surface realisers for each new domain and allow a higher reuse of resources by training realisers from human data.

[Barzilay and Lee \[2002\]](#) use multiple sequence alignment, a method from bioinformatics, to obtain lattices of surface form variants for a semantic concept in the domain of mathematical proofs. Their idea is based on a multi-parallel corpus that contains several realisations for a semantic concept. They apply the iterative

¹We assume that a PCFG can be treated as a graphical model since it uses graph-based probability distributions.

2. Graphical Models for Natural Language Generation

pair-wise alignment algorithm to learn a set of lattices, representing generation spaces for a semantic concept, from the corpus. In a human evaluation study that compared their system output with system output of a hand-crafted system for the same domain, results showed that humans judged both outputs comparably.

Georgila et al. [2002] present a bidirectional approach in which they use Hidden Markov Models, HMMs, for both speech recognition and language generation in a spoken dialogue system. The HMM-based approach helps them to, on the one hand, recognise ungrammatical or partial incoming user utterances, but at the same time, generate grammatical system output. They present an algorithm that is able to produce HMMs containing grammatical structure from human data through generalisations.

Belz [2008] presents Probabilistic Context-Free Representational Underspecified (*pCRU*) —a generation framework that uses probabilistic context-free grammars, learnt from a corpus, to generate variable texts in the domain of weather forecasts. The approach can be distinguished from conventional n-gram generators, which employ an over-generate and rank approach, by its property to inform the generation process statistically as it progresses. In an evaluation that compared several *pCRU* systems which differed in their way to make use of the statistical information, it was shown that the best *pCRU* system was able to generate texts that were rated more highly by human judges than a set of texts written by human experts on weather forecasts.

Mairesse et al. [2010] present BAGEL (Bayesian networks for generation using active learning), which is a system that uses Dynamic Bayesian Networks for surface realisation within an Active Learning framework. BAGEL learns surface realisation variants for a semantic concept from a semantically aligned and annotated corpus from the domain of restaurant recommendations. The system is shown to generate natural surface forms even under previously unseen circumstances due to its learning mechanism: the system will itself determine the next semantic frame to be annotated based on its current model. This will usually correspond to the frame for which there is most uncertainty. An evaluation with

2. RELATED WORK

human judges showed that the BAGEL output was rated close to the human gold standard in terms of its naturalness and informativeness.

[Konstas and Lapata \[2012\]](#) present generation from weighted hypergraphs that are induced from data using an unsupervised learning algorithm. The hypergraphs can represent exponentially many realisation candidates, which are ranked according to local and global features using discriminative re-ranking. The authors are thus able to combine both content selection and surface realisation into a single process and report improved generation performance over baselines.

[Dethlefs et al. \[2013\]](#) present an approach to surface realisation as a sequence labelling task using conditional random fields (CRFs). The authors define a mapping of (observed) semantic input variables onto a (hidden) sequence of surface realisations. Their domain of application is restaurant recommendations and the CRFs are automatically trained from a grammar induction algorithm. The authors report a comparison with several state-of-the-art baselines and find that using CRFs with features of the full utterance context performs better than considering the local context alone.

While this thesis follows the general idea of previous work of training graphical models based on human data for natural and variable surface realisation, it goes beyond previous work by offering a joint optimisation perspective under which surface realisation choices are considered directly in conjunction with content selection and utterance planning.

2.6 Conclusion

This chapter has provided an overview of research that is related to the work presented in this thesis. We have reviewed alternative approaches to context-sensitive generation, including rule-based generation, planning or constraint satisfaction-based generation, generation using trainable approaches such as supervised learning and several others, which are all summarised in Table [2.1](#). All of these meth-

2. Conclusion

ods have distinct advantages, such as the rich human or empirical knowledge that informs rule-based or planning approaches or the data-drivenness of supervised learning approaches. On the other hand, we have argued that rule-based and planning approaches tend to be expensive and time-consuming to design, especially for large and complex domains for which a fine-grained adaptation is required. While supervised learning approaches are typically cheaper and faster to develop, they can produce unpredictable behaviour under circumstances for which they were not trained. Based on these problems, we investigated previous

Authors	Domain	Method
Patten, 1988	expert systems	Constraint Satisfaction
Bateman and Paris, 1989	electric circuits	Rules
Paris, 1993	Technical devices	Rules
Bateman and Teich, 1995	biographies	Rules
Paris et al., 2004	web search	Rules
Foster and Oberlander, 2006	facial displays	Stochastic Knowledge
Stoia et al., 2006	virtual navigation	Decision Trees
Demberg and Moore, 2006	flight information	Clustering
Piwek and van Deemter, 2007	scripted dialogue	Constraint Satisfaction
Walker et al., 2007	restaurant information	Stochastic Knowledge
Garoufi and Koller, 2010	virtual navigation	Planning
Roth and Frank, 2010	outdoor navigation	EM-algorithm
Golland et al., 2010	spatial descriptions	game theory
White et al., 2010	flight information	Rules
Bouttaz et al., 2011	web tools descriptions	Rules based on permissions, prohibitions and obligations
Chiarcos, 2011	referring expressions	Salience measures
Dethlefs et al., 2011	outdoor navigation	Rules
Garoufi and Koller, 2011	virtual navigation	planning

Table 2.1: Summary of approaches towards context-sensitive generation, ordered chronologically with respect to authors, the domain addressed, the context variables involved and the method used.

work in applying reinforcement learning to language generation. Reinforcement learning has the benefit of being trained directly from human data and from human-based (simulated) interactions. An RL agent therefore learns directly through trial and error until its action policy converges to an optimal policy. Unfortunately, current approaches using RL for generation have only addressed

2. RELATED WORK

flat RL scenarios, which are affected by the *curse of dimensionality*, the property that their state-action space grows exponentially according to the number of its variables. This restricts flat RL to small-scale applications and makes them unsuitable for real-world systems. In addition, prior work in applying RL to generation has not taken a joint optimisation of NLG subtasks into account but has instead focused on decision making within a single NLG module.

To assess the potential benefits of a joint treatment of interrelated but distinct subtasks, we have reviewed related work on joint optimisations or treatments involving applications of natural language processing. Studies from the areas of NLG, dialogue management, route generation, speech synthesis and multi-modal output generation have all shown clear trends towards the benefits that can be gained through a joint treatment of subtasks. However, most approaches focus on two subtasks only for their joint treatment. While [Marciniak and Strube \[2005\]](#) represents an exception in dealing with eight subtasks simultaneously, all of these tasks are grammar-based and deal with the surface realisation of an utterance.

Finally, we have reviewed related work on using graphical models to assist surface realisation in an NLG system. This has included the use of probabilistic context-free grammars, lattices and Dynamic Bayesian Networks. Research in this area has shown that graphical models can support variable and natural surface realisation results that receive positive and near-human ratings from human judges.

The next chapters will proceed to develop a learning framework for situated NLG that is based on hierarchical reinforcement learning with a joint optimisation of NLG subtasks, a hierarchical information state for the specification of prior knowledge and graphical models as generation spaces for surface realisation.

Chapter 3

Hierarchical Reinforcement Learning for NLG

3.1 Introduction

Natural Language Generation decisions of content selection, utterance planning and surface realisation are in many ways interrelated and treating them in a joint fashion should represent one of the fundamental properties of generation in situated interaction. This claim represents the core hypothesis of this thesis whose foundations will be developed in this chapter. We will introduce a model for the joint optimisation of NLG decisions based on hierarchical reinforcement learning and present some first promising results in favour of this approach.

In order to establish the theoretical foundation of joint NLG for situated interaction, we will first review some of the intuitive ideas underlying reinforcement learning which are familiar from our daily lives. We will then proceed to introducing the computational framework of reinforcement learning based on the Markov Decision Process, or MDP. We will review important value functions and learning algorithms that can be used to learn optimal reinforcement learning policies for single learning agents. Finally, to exemplify the dynamics of this flat, single-policy, reinforcement learning setting, we will design an MDP for referring expression generation. All of this will be addressed in Section 3.2.

Unfortunately, flat reinforcement learning has a number of drawbacks for real-world applications or complex systems for situated interaction such as we wish to address. Since the state space of reinforcement learning agents grows exponentially according to the number of state variables, learning flat policies for large search spaces quickly becomes infeasible. Figure 3.1 illustrates the exponential growth problem graphically where the state grows according to the number of binary variables taken into account.

In this thesis, we will therefore suggest to use a *divide-and-conquer* approach

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

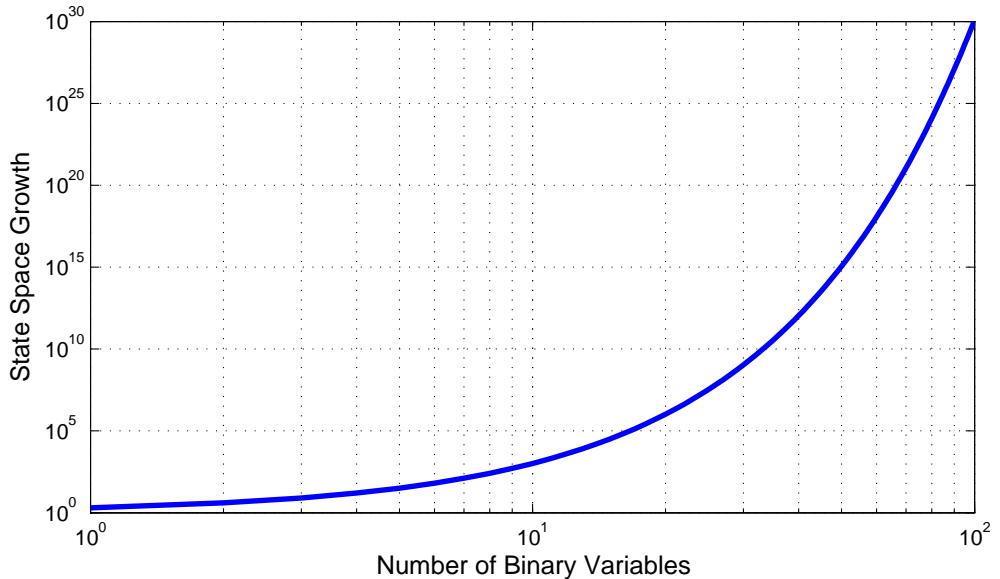


Figure 3.1: Illustration of the *curse of dimensionality* problem that reinforcement learning agents face. An agent’s state space grows exponentially according to the number of variables taken into account (here plotted in log space).

instead. It is based on a hierarchy of learning agents that address several sub-tasks of the overall, complex generation task. The model underlying hierarchical reinforcement learning is the Semi-Markov Decision Process, or SMDP, whose characteristic property is that actions may take a variable amount of time to complete and correspond to generation subtasks. Again, we will introduce the main value functions and an algorithm for hierarchical policy learning. We will exemplify the workings of an SMDP by applying it to the same referring expression generation task as the MDP before. Hierarchical reinforcement learning will be the topic of Section 3.3. With this background, we also present a comparison between reinforcement learning with and without a hierarchical setting and argue for the better suitability of the hierarchical case for realistic NLG systems. It scales to larger search spaces and is thus better able to account for the complex interdependencies that are a typical property of situated NLG systems. In particular, we find striking interdependencies among the NLG core tasks of con-

3. Reinforcement Learning

tent selection, utterance planning and surface realisation. Since the user needs to comprehend utterances online, while carrying out instructions, they should be maximally informative, i.e. contain as much detail as necessary, but not more. Similarly, we should generate as few utterances as possible to encourage short interactions. However, utterances should not be too long, because otherwise they become difficult to comprehend and memorise. Surface realisation should learn to distinguish grammatical and ungrammatical utterances, but at the same time, it should make decisions according to the user’s information need.

After having introduced and exemplified the core computational model of this thesis, we will use it to design a full NLG system for situated interaction including the tasks of referring expression generation, navigation instruction generation, and generation of high-level interaction behaviour such as giving feedback on the user’s progress or providing help for troubleshooting. We will design this system in Section 3.4 based on a set of annotated human-human interaction data. In Section 3.5, we will establish the experimental setting for training the learning agent and estimate a simulated environment from the annotated human data. A different set of human evaluation data will be used to induce a data-driven reward function for situated interaction. We will argue for a graded task success metric because it is more informative with respect to the difficulties users encounter during an interaction.

Finally, we will train our hierarchical learning agent and present a first set of results in Section 3.6. The results will support our foundational hypothesis that a joint optimisation of NLG core tasks achieves better performance than an optimisation in isolation which traditional NLG architectures have supported. We will argue that ignoring the ubiquitous interrelations between different NLG tasks means ultimately to leave unaddressed a fundamental property of situated interaction. We will then draw some conclusions in Section 3.7 and discuss remaining limitations of our approach that following chapters will address.

3.2 Reinforcement Learning

The idea of learning through trial and error is a familiar concept in the life of human beings. In our early years, we apply it when we learn to walk or

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

learn to speak. Later, we may learn to ride a bike, learn to swim or learn to spell. Eventually, we may need to learn to maintain a complex network of social relationships in order to succeed in our personal and professional lives. What all of these examples have in common is that they require a notion of practice to reach *optimal performance* and continuous interaction with the *environment*. The environment refers to the external world including physical objects and other people, that is everything that we, as *learning agents*, have no direct control over. While we practice, the environment provides continuous feedback on how well we are doing in the form of positive or negative *reinforcement signals*. Positive reinforcement signals can come in multiple forms, such as a piece of chocolate, cycling a few metres without stabilisers, receiving a promotion or higher salary or simply a feeling of self-satisfaction. Negative reinforcement signals similarly can be physical pain, a reprimand or becoming unemployed. We will use the term *reward* here to refer to both positive and negative reinforcement signals as real numbers. Every learning agent's goal in the long run is to learn to act upon the environment so as to change it favourably towards their own goals, so that the highest possible rewards are obtained. In other words, an agent needs to analyse the environment, decide on the best action to take and observe the reward the action yields. The environment itself is changed by any action the agent takes.

In this chapter, we will apply the theory of learning through interaction with the environment to natural language generation. As in many other domains, NLG systems face a number of problems whose solutions are unclear even to the system designer. In a referring expression generation task, for example, where a referent has multiple distinguishing characteristics (colour, form, size, location) and multiple distractors (other buttons) or nearby landmarks, which characteristics, distractors or landmarks will be most suitable to uniquely identify the referent? This thesis will argue that a good way to discover the best strategy is to explore different ones in interactions with human interlocutors and see which strategies humans prefer.

In any learning domain, rewards only inform the agent about *what* to do, not *how* to do it. It therefore has to try different strategies in the same situation to discover which of those strategies works best, i.e. yields the best long-term reward, also known as the *expected return*. The agent therefore faces a trade-

3. Reinforcement Learning

off between *exploration* and *exploitation*: in order to obtain high rewards, it needs to exploit actions that have yielded good returns in the past; on the other hand, it needs to try previously unexplored actions to discover whether they yield even better returns in the long run. Depending on the task, long-term, delayed rewards often yield better returns than immediate rewards and are therefore more desirable to the learning agent. Again this bears some similarities to real world examples. Learning to ride a bike may initially inflict a number of bruises and scratches, but will lead to a quick and comfortable way of traveling in the long run. Similarly, pursuing higher education typically yields more rewards in the long-term than in the short-term. Identifying a referent by a simple property such as colour may initially require clarification, but may in the longer run allow more efficient referring expressions.

The concepts of exploration and exploitation and delayed rewards will be the most important ingredients to the computational model of reinforcement learning that we will develop in the remainder of this chapter. Please see [Skinner \[1938\]](#) for an early psychological and educational view on reinforcement learning.

3.2.1 The Markov Decision Process

The goal of an RL agent is to map situations to actions in a goal-directed manner so as to maximise a long-term, numeric reward signal. The computational model underlying reinforcement learning agents is the *Markov Decision Process*, or MDP. An MDP can be defined formally as a four-tuple $\langle S, A, T, R \rangle$.

- $S = \{s_0, s_1, s_2, \dots, s_N\}$ is a set of environment states that summarise all information, present and past, that the agent needs to make optimal decisions. In referring expression generation this includes all the information that is required to generate a successful referring expression, such as information on the referent's properties and the properties of its distractors. Environment states need to fulfill the *Markov property* which requires that the current state captures information in such a way that an agent making decisions only based on the current state is just as successful as if it made decisions based on the entire history of states (hence the name MDP). In addition, states must allow the agent to monitor its progress in the learn-

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

ing task at any time and observe the effects that its actions have. Thus whenever the agent takes an action a in state s at time step t , the updated state $s_{t+1} = s'$ (at time step $t + 1$) should contain the action's effect on the environment. In this way, the agent is able to learn from its experience.

- $A = \{a_0, a_1, a_2, \dots, a_M\}$ is the set of actions available to the agent. It defines the agent's behavioural potential and forms the basis for decision making and the principle of exploration and exploitation. For referring expression generation, it may include the repertoire of choices the agent faces in generating a referring expression (mention the referent's colour, locate the referent in relation to a landmark, etc.).
- T is a probabilistic transition function indicating the next state s' from the current state s and the action a . For instance having generated a particular referring expression in the current spatial environment, how does the referring expression change the environment? T is represented by a conditional probability distribution $P(s'|s, a)$ satisfying $\sum_{s' \in S} P(s'|s, a) = 1, \forall (s, a)$.
- R is a reward function $R(s'|s, a)$ specifying a numeric reward that an agent receives for taking action a in state s . Rewards allow the agent to evaluate its decision making process (how good a choice was it to identify a referent only by its colour when several objects of the same colour are present?). The reward at time $t + 1$ is also denoted as r' .

The dynamics of an MDP can be described as follows. At the beginning of an interaction between the agent and the environment, when the time step $t = 0$, the agent receives a representation of the environment, called the state $s_t \in S$. It needs to perform an action $a_t \in A$, that is the set of actions available in state s_t . As a result, the agent will receive a reward $r_{t+1} \in R$ and observe the next state $s_{t+1} \in S$, which is the updated environment state. This process can be seen as a finite sequence of states, actions and rewards $\{s_0, a_0, r_1, s_1, a_1, \dots, r_{t-1}, s_t\}$ and is illustrated in Figure 3.2. Any mapping from states to actions is called a *policy* π . Ultimately, the agent's goal is to learn an *optimal policy* denoted as π^* , a mapping from every state s to an action a that will yield the highest expected return. In

3. Reinforcement Learning

particular, we wish to develop an agent that learns an *optimal generation policy*, i.e. a mapping from any generation context s to a generation action a that will maximise the expected success of a generated utterance.

Remember that an agent does not learn a policy to optimise immediate rewards. Rather, the agent's goal is to optimise the cumulative reward that can be obtained for an entire sequence of actions. Thus, to identify those actions that yield the best expected long-term return, the agent needs to apply a trial and error strategy trying individual actions many times in different states in order to estimate their real reward. For example, for a sequence of rewards $r_{t+1}, r_{t+2}, r_{t+3}, \dots, r_T$, where r_T is the final time step, we can define the expected return as the sum of single rewards $v_t = r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_T$ whenever we are dealing with an *episodic learning task*, i.e. whenever a task breaks up naturally into a finite sequence of time steps. We can call sequences of such time steps *episodes*. Other learning tasks, which do not have obvious final steps, are called *continuing tasks*. Some authors also call these tasks *finite horizon* and *infinite horizon* models [Dietterich, 2000c; Kaelbling et al., 1996]. This thesis will focus on episodic tasks. In order to give the agent an indication of the value of future rewards in relation to immediate rewards, we assign cumulative *discounted rewards* according to

$$v_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^\tau r_{t+\tau} = \sum_{k=0}^{\tau-1} \gamma^k r_{t+k+1}, \quad (3.1)$$

where $t + \tau$ is the total number of time steps T and γ is called the *discount rate* $0 \leq \gamma \leq 1$. “[A] reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately” [Sutton and Barto, 1998], p. 58, according to this equation. If γ is 0, the agent will be concerned only with maximising immediate rewards. As γ approaches 1, both immediate and future rewards will be increasingly equally valuable.

3.2.2 Policy Learning

Above we defined a policy π as a mapping from (generation) states to (generation) actions. Accordingly, we defined a mapping from states to actions so as to obtain

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

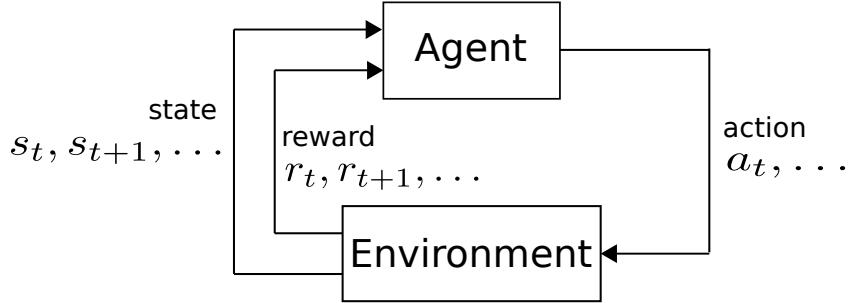


Figure 3.2: Illustration of the agent-environment interaction (adapted from Sutton and Barto [1998], page 52).

maximal rewards as an optimal policy π^* . An agent can learn multiple optimal policies as long as they yield the same expected return. In the following, we will describe how an optimal policy is defined formally and can be estimated during learning.

3.2.2.1 Estimating Value Functions

We can determine the expected return of a policy by estimating its *state-value function* and its *action-value function*. The state-value function $V^\pi(s)$ of a policy π indicates the expected return of starting in state s and then following π . We can express this formally as

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi \left\{ \sum_{k=0}^{\tau-1} \gamma^k r_{t+k+1} | s_t = s \right\}, \quad (3.2)$$

where $E_\pi\{\}$ denotes the expected return. The optimal state-value function is

$$V^*(s) = \max_\pi V^\pi(s), \quad (3.3)$$

which yields the best expected return for each visited state. The action-value function Q^π of a policy π is the expected return for starting in state s , taking

3. Reinforcement Learning

action a and then following π . It is defined as

$$Q^\pi(s, a) = E_\pi \{R_t | s_t = s, a_t = a\} = E_\pi \left\{ \sum_{k=0}^{\tau-1} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\}. \quad (3.4)$$

The optimal action-value function that yields the best expected return for each action a taken in state s is defined as

$$Q^*(s, a) = \max_\pi Q^\pi(s, a). \quad (3.5)$$

All optimal policies, in case there is more than one, share $V^*(s)$ and $Q^*(s, a)$, the optimal state-value and action-value functions. These value functions are estimated during the agent's learning process. Assuming that an average value is kept for each state s that is visited and for each action a following state s , the state and action values will be re-estimated at each encounter and eventually converge to their actual values as the agent tries them infinitely often. The optimal value function will appear from this process automatically. Specifically, the optimal state-value function for an optimal policy π^* can be discovered recursively by solving the Bellman equation for $V^*(s)$

$$V^*(s) = \max_{a \in A} Q^{\pi^*}(s, a) = \max_a \sum_{s'} P(s'|s, a) \left[R(s'|s, a) + \gamma V^*(s') \right]. \quad (3.6)$$

Similarly, an optimal action-value function for an optimal policy π^* can be discovered recursively by solving the Bellman equation for $Q^*(s, a)$

$$Q^*(s, a) = E \left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right\} \quad (3.7)$$

$$= \sum_{s'} P(s'|s, a) \left[R(s'|s, a) + \gamma \max_{a'} Q^*(s', a') \right]. \quad (3.8)$$

While the Bellman equation yields exact results for each value function, it is not usually practically possible to determine value functions exactly due to

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

the large amount of computation involved in moderate or large learning tasks (such as we are faced with for most NLG tasks). Instead, a frequent solution is to compute approximations of the real values and base decision making on these approximations. Another alternative to estimating value functions entirely is to use function approximation techniques. In this thesis, we will focus on approximating value functions, but see [Sutton and Barto \[1998\]](#), Chapter 8, for an overview of function approximation methods.

Since value functions can only be computed reliably if the agent explores them sufficiently often, one question is how to best address the important trade-off between exploration and exploitation in practice. The agent needs to perform enough exploration to estimate value function reliably, but it also needs to increasingly prefer actions that yield good return in order to obtain higher rewards over time. There are three popular methods to address the agent's exploration-exploitation dilemma. The simplest of them is called the *sample-average* method, or *greedy* method since it always exploits those actions that have yielded the highest rewards in the past. It computes the value of an action as a mean of the reward received when selecting the action according to

$$Q_t(a) = \frac{r_1 + r_2 + \dots + r_{k_a}}{k_a}, \quad (3.9)$$

where $Q_t(a)$ is the estimated value of action a at time t that it was selected and k is the number of previous times that a was selected. An alternative to greedy action selection is the ϵ -*greedy* method, which behaves greedily most of the time, but selects a random other action for exploration ϵ percent of the time. This has the advantage that *each* action will be explored an infinite number of times as the agent's learning process continues and action values converge towards their real values. Finally, the *softmax* method for action selection ranks actions according to their expected return and then prefers to explore higher ranked actions over lower ranked ones. In this way, the agent is more likely to explore the next-to-best action than to explore the worst action. This is not true for the greedy methods. Which of these action selection methods works best depends on the learning task. Importantly, using softmax requires that the system designer has prior knowledge about the value of different actions. In this thesis, we do not assume such prior

3. Reinforcement Learning

Algorithm 1 The Q-Learning Algorithm.

```

1: function Q-LEARNING(states  $S$ , actions  $A$ , transitions  $T$ , rewards  $R$ , discount  $\gamma$ )
2:   Initialise  $Q(s, a)$  arbitrarily and initialise  $\alpha$  to 1
3:   for each episode do:
4:     Initialise  $s$ 
5:     while  $s$  is not a terminal state do:
6:       Choose  $a$  from  $s$  using policy derived form  $Q$  (e.g.,  $\epsilon$ -greedy)
7:       Take action  $a$ 
8:       Observe  $r$  from  $R$ 
9:       Observe  $s'$  from  $T$ 
10:      Decay  $\alpha$  (e.g.  $\alpha = 1/(1 + \text{visits}(s, a))$ )
11:       $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
12:       $s \leftarrow s'$ 
13:    end while
14:   end for
15:   return  $Q(s, a)$ 
16: end function

```

knowledge and will use the ϵ -greedy method.

3.2.2.2 The Q-Learning Algorithm

With all the prerequisites in place, we are now ready to bring all components together into a full algorithm which allows the agent to learn from its experience during the learning process. In particular, we will from now on focus on learning the action-value function $Q^\pi(s, a)$ for each policy, i.e. the expected return of state-action pair (s, a) . In other words, for an agent that makes generation choice a in the context of s and then follows the generation policy π , what is the expected return? Algorithm 1 shows the Q-Learning algorithm [Watkins, 1989], which computes Q -values according to

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)], \quad (3.10)$$

where α is a *step-size parameter* which indicates the learning rate that decays from 1 to 0, for example as in $\alpha = 1/(1 + \text{visits}(s, a))$, where $\text{visits}(s, a)$ corresponds to the number of times that the state-action pair (s, a) has been visited

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

previous to time step t . An optimal learnt policy π^* is then defined by

$$\pi^*(s) = \arg \max_{a \in A} Q^*(s, a), \quad (3.11)$$

where again the Q -function specifies the cumulative reward of starting in state s , taking action a and then following policy π^* . Initial Q -values are typically 0 assuming no prior knowledge but can be set optimistically depending on the learning task. Q-Learning is an *off-policy* algorithm for learning a policy. We call this algorithm off-policy, because it distinguishes the *behaviour policy*, the policy it currently uses for decision making, from the *estimation policy*, the policy it is currently aiming to improve. This means that the learned action-value function $Q(s, a)$ directly approximates the optimal action-value function $Q^*(s, a)$ independently of the policy currently followed. Q-Learning converges to the optimal Q -function Q^* with probability 1 under the assumption that all state-action pairs continue to be updated infinitely often [Jaakkola et al., 1994; Tsitsiklis, 1994; Watkins, 1989]. Off-policy algorithms stand in contrast to *on-policy* algorithms, such as Sarsa [Rummery and Niranjan, 1994; Sutton, 1996], in which the behaviour and the estimation policy are the same. Sarsa also converges to optimality.

Both Q-Learning and Sarsa fall into a category of learning algorithms called *temporal-difference learning*. They can be contrasted with *dynamic programming* methods and *Monte Carlo* methods. Dynamic programming algorithms are usually not practically feasible due to their property of computing values exactly and assuming a complete model of the environment. Monte Carlo methods allow learning to occur only at the very end of a learning episode, which can slow learning for long episodes. In contrast, temporal-difference learning allows the agent to learn from every transition it makes. In this thesis, we will concentrate on temporal-difference learning. See Sutton and Barto [1998], Chapter 4, for an overview of dynamic programming methods and Chapter 5 for an overview of Monte Carlo methods. Further alternatives to Q-Learning are *actor-critic* methods or R-Learning (see *ibid.*, Chapter 6). Further, all of these algorithms could be equipped with *eligibility traces* which mark each state-action pair that they visit with a trace indicating its eligibility to undergo learning. See *ibid.*, Chapter 7, for

3. Reinforcement Learning

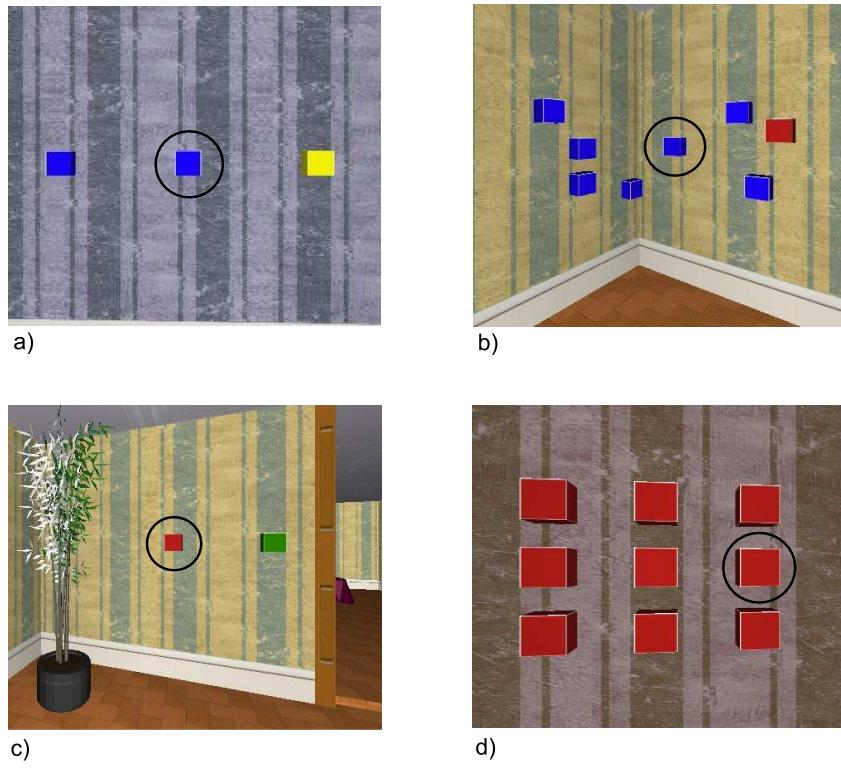


Figure 3.3: Example scenes for referring expression generation: scenes contain a referent (circled) that needs to be uniquely identified with respect to its distractors (other buttons) and landmarks in the environment.

an overview of eligibility traces. We will not pursue them further in this thesis. Also, see *ibid.*, Chapter 9, for model-based approaches to reinforcement learning. For a discussion of the computational complexity of solving MDPs, please see [Littman et al. \[1995\]](#); [Papadimitriou and Tsitsiklis \[1987\]](#).

3.2.3 An MDP for Referring Expression Generation

After having introduced the most important aspects of flat reinforcement learning and its underlying computational model, the MDP, let us design an MDP for the task that we have been using as an example all through this section: referring expression generation (REG).

Imagine you had the task of designing an REG module to uniquely identify

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

Feature	Values	Description
f_0	true,false	Is the referent's colour discriminating?
f_1	none,one,few,many	How many distractors are present?
f_2	none,one,few,many	How many landmarks are present?
f_3	true,false	Is the referent's spatial position discriminating?
f_4	null,yes,no	Has colour been mentioned?
f_5	null,yes,no	Has a distractor been mentioned?
f_6	null,yes,no	Has a landmark been mentioned?
f_7	null,yes,no	Has a spatial position been mentioned?

Action	Description
a_0	Mention referent colour
a_1	Do not mention referent colour
a_2	Include distractor
a_3	Do not include distractor
a_4	Include landmark
a_5	Do not include landmark
a_6	Include spatial position
a_7	Do not include spatial position

Figure 3.4: State and action sets for a flat reinforcement learning agent for referring expression generation. The state-action space of this agent is $2^2 \times 3^4 \times 4^2 \times 8 = 41472$.

referents in a situated setting such as exemplified in Figure 3.3. In all of these scenes, we are supposed to identify one of the depicted buttons and generate a referring expression instructing a user to press the intended button. The information we have available about the situation is whether the referent's colour is discriminating or not, the number of present distractors (objects of the same type as the referent, but which are not the referent), the number of present landmarks, and whether or not the referent's spatial position (left, right, top, etc.) is discriminating or not. This information is provided by the environment. In addition, each action that the agent takes will be represented as whether or not the referent's colour has been mentioned, whether or not a distractor has been mentioned, whether or not a landmark has been mentioned and whether or not

3. Hierarchical Reinforcement Learning

a spatial relation has been mentioned. The conjunction of these features makes up the learning agent’s state representation. The agent’s behavioural potential is defined by its action set. It contains the following choices: mention the referent’s colour, do not mention the referent’s colour, include a distractor, do not include a distractor, mention a landmark, do not mention a landmark, include a spatial relation, do not include a spatial relation. After each action, the transition function will update state s to the next state s' to reflect the action that was taken. See Figure 3.4 for a summary of the outlined states and actions (states are represented as features $f_0 \dots f_7$ and actions as $a_0 \dots a_7$). Based on the state and action spaces shown in Figure 3.4, our agent’s state space is the combination of feature values f_i , and the agent’s action set $A = \{a_j\}$. The size of the flat state-action space is $|S \times A| = \sum_i |f_i| \times |A| = 41472$. As a reward function, we will assign a reward of +1 for reaching the goal state and having generated a uniquely identifying referring expression. We will assign a reward of 0 for reaching the goal state and not having generated a referring expression that is uniquely identifying. Finally, a reward of -1 will be given for any other action to prevent the agent from entering into loops by choosing actions multiple times. With the four main components of an MDP defined, the state set S , the action set A , the transition function T (specified from prior knowledge) and the reward function R , we could use the Q-Learning algorithm to learn an optimal policy for referring expression generation in a situated setting. We will now discuss reinforcement learning with a hierarchical setting and then compare our MDP with an SMDP for the same referring expression generation task.

3.3 Hierarchical Reinforcement Learning

Flat reinforcement learning is a powerful method when applied to small state spaces such as a simple referring expression generation task with a limited set of choices. Unfortunately, it is affected by *the curse of dimensionality*, the fact that an agent’s state space grows exponentially to the number of state variables considered. Thus, the more information we consider to make decisions, the harder the task will become for a learning agent. Learning an optimal policy for a realistic task within a feasible time frame (given sufficient memory) can become impossible

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

under such circumstances. Hierarchical reinforcement learning represents one way to mitigate this problem by offering a divide-and-conquer approach. By decomposing a large learning task into a hierarchy of smaller learning tasks, complexity is reduced and the RL agent is enabled to scale to large and complex learning tasks. The intuition underlying hierarchical RL is that a system designer will be able to identify a set of subtasks among the ultimate learning task. An example given by [Barto and Mahadevan \[2003\]](#) is a robot that navigates in an office environment. One of the robot’s subtasks is to open a door, which is characterised by the primitive actions of reaching, grasping and turning a door knob. Similarly, in generation we can divide the task of generating a referring expression into the subtasks of describing the referent, describing the distractor, describing an adjacent landmark, etc. In this way, the original complexity of the task is reduced and learning larger tasks becomes feasible.

A hierarchical learnt policy has been shown to be *recursively optimal* or *locally optimal* (in contrast to globally optimal policies not treated here): the policy of each subtask is optimal given the policies of its children. An advantage of local optimality is that each subtask can be treated in isolation of the context in which it is executed. A further advantage of hierarchical RL and its local optimality is that it facilitates the reuse of behaviours so that a (generation) behaviour learnt as part of one policy can be transferred to another.

3.3.1 The Semi-Markov Decision Process

Any flat learning agent \mathcal{M} that is characterised by a single MDP can be decomposed into a set of subtasks M_j^i where i and j are indices that uniquely identify each subtask in a hierarchy of subtasks such that $\mathcal{M} = \{M_0^0, M_0^1, M_1^1, M_2^1, \dots, M_M^N\}$. Note that these indices do not specify the order of execution of subtasks. The order of execution represents one of the agent’s learning tasks. Each subtask is defined as a Semi-Markov Decision Process (or SMDP) $M_j^i = \langle S_j^i, A_j^i, T_j^i, R_j^i \rangle$.

- $S_j^i = \{s_0, s_1, s_2, \dots, s_N\}$ is a set of states of subtask M_j^i that fulfill the Markov property. They represent a formalisation of the learning environment for a subtask such as the context of a generation situation.

3. Hierarchical Reinforcement Learning

- $A_j^i = \{a_0, a_1, a_2, \dots, a_M\}$ is a set of actions of subtask M_j^i that can be either primitive or composite. Primitive actions are single-step actions as in an MDP and receive single rewards. Composite actions are temporally-extended actions that correspond to other subtasks in the hierarchy which are children of the parent subtask. They receive cumulative rewards. The execution of a composite action, or subtask, takes a variable number of time steps to complete. This is indicated by the random variable τ , and it represents the temporal abstraction that is characteristic of the SMDP model (and that distinguishes it from an MDP). The parent SMDP of a subtask remains in its current state s until control is transferred back to it. It then makes a transition to the next state s' .
- T_j^i is a probabilistic transition function of subtask M_j^i indicating the next state s' from the current state s and the action a . It is represented by a conditional probability distribution $P_j^i(s', \tau | s, a)$ satisfying $\sum_{s', \tau \in S_j^i} P_j^i(s', \tau | s, a) = 1, \forall (s, a)$. The parameter τ represents the number of time steps a subtask, such as a generation subroutine, takes to complete.
- R_j^i is a reward function $R_j^i(s', \tau | s, a)$ for subtask M_j^i that specifies the reward the agent receives for taking action a (lasting τ time steps) in state s . Discounted cumulative rewards in a hierarchy of SMDPs can be computed according to

$$v_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{\tau-1} r_{t+\tau}, \quad (3.12)$$

which corresponds to the cumulative reward that was collected during the execution of action a , which can either be a primitive or a composite action.

The hierarchy of subtasks \mathcal{M} that results from decomposing an MDP into a set of SMDPs is characterised by a single root subtask M_0^0 and multiple children subtasks $M_j^{i>0}$ so that solving M_0^0 means solving \mathcal{M} . Each subtask can only invoke its immediate children, subtasks cannot invoke themselves either directly or indirectly. Figure 3.5 shows an illustration of the dynamics of an SMDP. Figure 3.6 shows a hierarchy of SMDPs. The policies for a hierarchy of SMDPs, which could

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

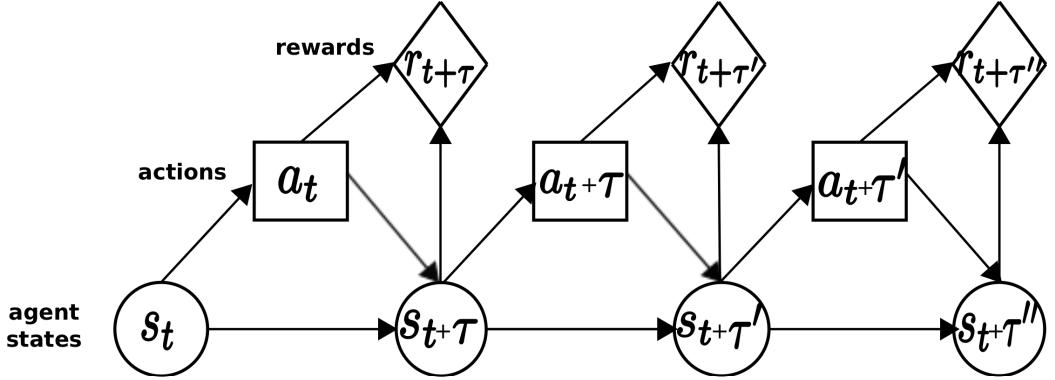


Figure 3.5: Dynamics of a Semi-Markov Decision Process or SMDP [Cuayáhuitl, 2009], Chapter 3. State transitions and rewards depend on the number of time steps, denoted τ , actions take to complete.

correspond to a hierarchy of generation subtasks, can be learnt simultaneously using the MAXQ framework [Dietterich, 2000b,c]. We can define a hierarchical policy as $\pi = \{\pi_0^0, \pi_0^1, \pi_1^1, \pi_2^1, \dots\}$, where π_j^i is the policy of subtask M_j^i .

3.3.2 Policy Learning

A hierarchical policy π can be seen as a hierarchy of policies, where π_j^i denotes the policy of subtask M_j^i . A hierarchy is recursively optimal if each policy is optimal with regard to its children [Barto and Mahadevan, 2003].

3.3.2.1 Estimating Value Functions

Just as for the flat reinforcement learning setting, we can compute the state-value functions and action-value functions for each subtask in the hierarchy.

The state-value function of subtask M_j^i , $V_j^{\pi^i}(s)$, for starting in state s and then following policy π_j^i can be computed according to

$$V_j^{\pi^i}(s \in S_j^i) = E_\pi\{R_t | s_t = s\} = E_\pi \left\{ \sum_{\tau=0}^{\tau-1} \gamma^\tau r_{t+\tau+1} | s_t = s \right\}, \quad (3.13)$$

3. Hierarchical Reinforcement Learning

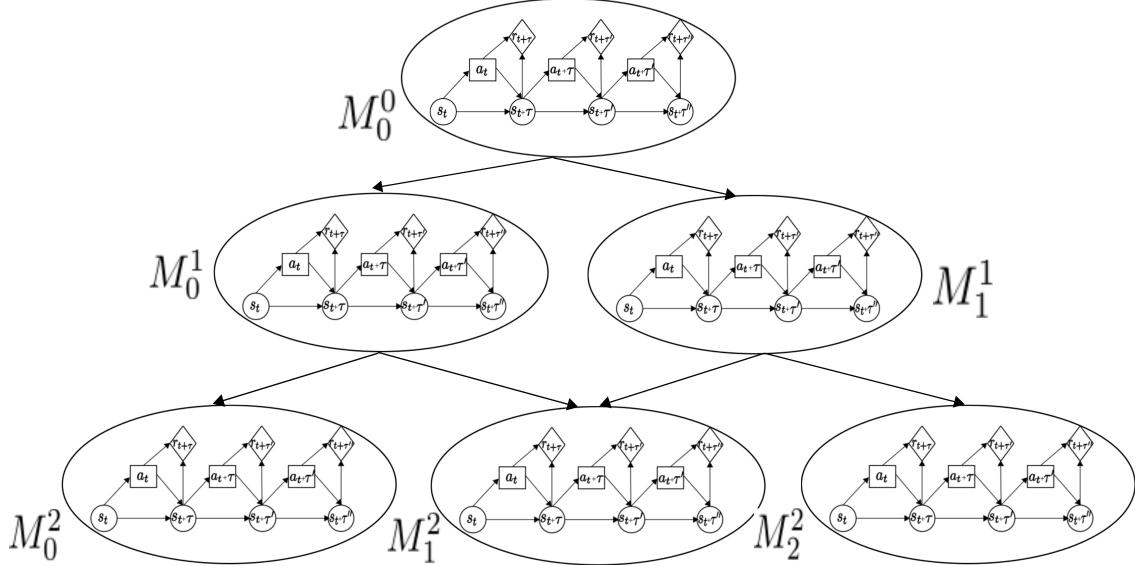


Figure 3.6: A hierarchy of Semi-MDPs where parent subtasks can call child subtasks and arrows indicate the flow of execution. The indices i and j serve only to uniquely identify an agent in the hierarchy, they do not specify an execution sequence (which is subject to optimisation).

where the optimal state-value function is

$$V_j^{*i}(s) = \max_{\pi_j^i} V_j^{\pi^i}(s). \quad (3.14)$$

The Bellman equation for the optimal state-value function can be expressed as

$$V_j^{*i}(s \in S_j^i) = \max_{a \in A_j^i} \left[\sum_{s', \tau} P_j^i(s', \tau | s, a) [R_j^i(s', \tau | s, a) + \gamma^\tau V_j^{*i}(s')] \right]. \quad (3.15)$$

Similarly, the action-value function $Q_j^{\pi^i}(s, a)$ of subtask M_j^i for starting in

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

state s , taking action a and then following policy π_j^i thereafter is defined by

$$Q^{\pi_j^i}(s \in S_j^i, a \in A_j^i) = E_{\pi} \{ R_t | s_t = s, a_t = a \} = E_{\pi} \left\{ \sum_{\tau=0}^{\tau-1} \gamma^{\tau} r_{t+\tau+1} | s_t = s, a_t = a \right\} \quad (3.16)$$

with the optimal action-value function

$$Q_j^{*i}(s, a) = \max_{\pi_j^i} Q_j^{\pi_j^i}(s, a). \quad (3.17)$$

The Bellman equation for the optimal action-value function is defined by

$$Q_j^{*i}(s, a) = \sum_{s', \tau} P_j^i(s', \tau | s, a) \left[R_j^i(s', \tau | s, a) + \gamma^{\tau} \max_{a'} Q_j^{*i}(s', a') \right]. \quad (3.18)$$

3.3.2.2 The HSMQ-Learning Algorithm

Algorithm 2 shows the Hierarchical Semi-Markov Q-Learning (or HSMQ-Learning) algorithm [Dietterich, 2000a], a form of hierarchical Q-Learning. Note the distinction here between a knowledge-rich state k , also referred to as knowledge base, and a knowledge-compact state s . The knowledge-rich state k refers to the collection of all state variables in the hierarchy \mathcal{M} , whereas the knowledge-compact state s refers only to the state variables that are relevant for decision making in a particular subtask M_j^i . The knowledge-rich state s is obtained from the knowledge-compact state k at the beginning of each execution cycle of the HSMQ-Learning algorithm. This process of reducing the full state space to those states relevant for decision making is called *state abstraction* [Dietterich, 2000b] and is one of the fundamental properties of hierarchical RL. It is one of the main solutions to address the curse of dimensionality mentioned earlier. The algorithm works by calling itself recursively to traverse downwards through the hierarchy until it eventually executes primitive actions in the leaf subtasks. This process is implemented using a stack mechanism where subtasks are pushed onto a stack when they are called and are popped off again when they reach their terminal state, i.e. terminate execution. This process continues until the stack is empty.

3. Hierarchical Reinforcement Learning

Algorithm 2 The HSMQ-Learning Algorithm.

```
function HSMQ(KnowledgeBase  $k$ , subtask  $M_j^i$ )
2:    $s \leftarrow$  knowledge-compact state in  $S_j^i$  initialised from  $k$ 
    $totalReward \leftarrow 0$ ,  $discount \leftarrow 1$ 
4:   while  $s$  is not a terminal state do
   Choose action  $a$  from  $s$  using policy derived from  $Q_j^i$  (e.g.  $\epsilon$ -greedy)
6:   Execute action  $a$  and update knowledge-rich state  $k$ 
   if  $a$  is primitive then
8:     Observe one-step reward  $r$ 
   else if  $a$  is composite then
10:     $r \leftarrow$  HSMQ( $k$ , model of composite action  $a$ )
   end if
12:     $totalReward \leftarrow totalReward + discount \times r$ 
     $discount \leftarrow discount \times \gamma$ 
14:    Observe resulting state  $s'$ 
    
$$Q_j^i(s, a) \leftarrow Q_j^i(s, a) + \alpha \left[ r + discount \times \max_{a' \in A_j^i} Q_j^i(s', a') - Q_j^i(s, a) \right]$$

16:     $s \leftarrow s'$ 
   end while
18:   return  $totalReward$ 
end function
```

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

Whenever a primitive, single-step, action is executed, only a single reward is observed. If an action is composite and corresponds to a subtask, the subtask is called and rewards are accumulated until it terminates. When control is transferred back to the calling subtask, a cumulative reward is received. Q -values in HSMQ-Learning are computed according to

$$Q_j^i(s, a) \leftarrow Q_j^i(s, a) + \alpha \left[r + \gamma^\tau \times \max_{a' \in A_j^i} Q_j^i(s', a') - Q_j^i(s, a) \right]. \quad (3.19)$$

Again, the optimal learnt hierarchical policy is defined by

$$\pi_j^{*i}(s) = \arg \max_{a \in A_j^i} Q_j^{*i}(s, a). \quad (3.20)$$

If we imagine that a generation task \mathcal{M} was decomposed into a number of generation subtasks $M_0^0 \dots M_M^N$, the HSMQ-Learning algorithm can be used to learn a hierarchical generation policy π . In this thesis, we will concentrate on hierarchical reinforcement learning for context-independent policies and with non-decomposed value functions. Please refer to [Barto and Mahadevan \[2003\]](#); [Cuayáhuitl \[2009\]](#); [Dietterich \[2000b\]](#) for an overview of other variants of hierarchical reinforcement learning.

3.3.3 An SMDP for Referring Expression Generation

Let us now compare the MDP we designed earlier for the task of referring expression generation with a hierarchically decomposed model for the same task: a hierarchy of SMDPs. To this end, we decompose the single task of generating referring expressions into four subtasks: making decisions concerning the referent, making decisions concerning the distractors, the landmarks, and the spatial relations. Figure 3.7 (on the bottom) shows the resulting hierarchy with the new distribution of state variables and actions across the root and subtasks. To ease the comparison, the flat definition is repeated on the top of the figure. While the flat learning agent had a state-action space of 41472, the hierarchical agent has a state-action space of only 420. This corresponds to roughly 2% of the original size. This huge reduction is indicative of one of the main advantages of hierar-

3. A Joint Learning Agent for Situated Interaction

chical reinforcement learning using a divide-and-conquer approach for addressing NLG tasks. Given the reduced state-action space of the hierarchical setting, we can speculate that learning times would be reduced and learning complex policies would accordingly be facilitated.

3.4 A Joint Learning Agent for Situated Interaction

In order to test the applicability of hierarchical reinforcement learning to real-world systems that support users in a full situated interaction task, let us add further complication to the earlier discussed referring expression generation task. In this section, we will introduce a hierarchical learning agent for the GIVE task, an interactive task that requires the generation of referring expressions, navigation instructions and a range of other behaviours such as confirming or correcting the user’s behaviour and reacting in a number of flexible ways to situations in which troubleshooting is needed. A special characteristic of the situated domain is the way in which content selection, utterance planning and surface realisation are interrelated. Content selection, for example, must not just aim to provide the user with enough information, but must also learn to avoid redundant detail in order to ease the user’s cognitive load. Similarly, utterance planning must be driven by generating the fewest possible utterances, but not at the cost of producing too long, and incomprehensible utterances. Surface realisation must learn to distinguish grammatical from ungrammatical surface forms, but also correspond to the user’s information need and degree of confusion. We will therefore aim for a unified treatment of decisions of all three subtasks so that, computationally speaking, we will compute cumulative rewards for all NLG subtasks rather than treating them as independent components.

We will design a hierarchical learning agent to address these trade-offs based on human interaction data that was collected in a Wizard-of-Oz setting. This data will inform the design of the agent, its behavioural potential and the simulated environment in which it will be trained. Further, we will induce a reward function from data collected in a human evaluation study of a real-world wayfind-

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

(a) Flat state-action space

Feature	Values	Description
f_0	true,false	Is the referent's colour discriminating?
f_1	none,one,few,many	How many distractors are present?
f_2	none,one,few,many	How many landmarks are present?
f_3	true,false	Is the referent's spatial position discriminating?
f_4	null,yes,no	Has colour been mentioned?
f_5	null,yes,no	Has a distractor been mentioned?
f_6	null,yes,no	Has a landmark been mentioned?
f_7	null,yes,no	Has a spatial position been mentioned?

Action	Description
a_0	Mention referent colour
a_1	Do not mention referent colour
a_2	Include distractor
a_3	Do not include distractor
a_4	Include landmark
a_5	Do not include landmark
a_6	Include spatial position
a_7	Do not include spatial position

(b) Hierarchical state-action space

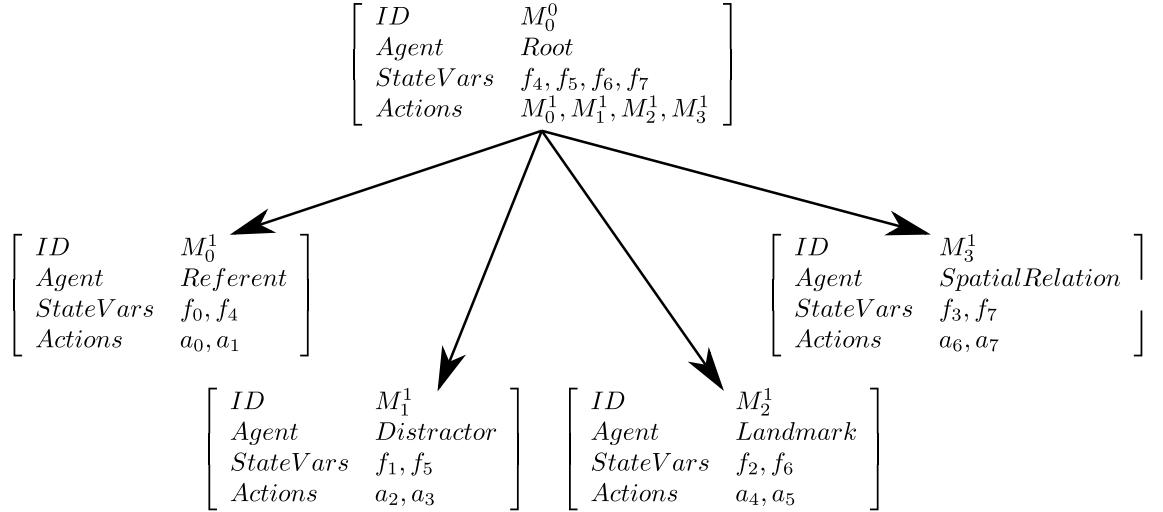


Figure 3.7: Flat and hierarchical state-action spaces for a simple learning agent for referring expression generation. Whilst the size of the flat agent has 41472 state-actions, the size of the hierarchical agent has only 420 state-actions. In the hierarchical case, a parent subtask can invoke child subtasks. When they terminate, control is returned to the caller.

3. A Joint Learning Agent for Situated Interaction

ing system. We will then train the agent and test the fundamental hypothesis of this thesis: that a jointly optimised policy of the NLG core tasks content selection, utterance planning and surface realisation will outperform a policy optimised in isolation.

3.4.1 The Domain: Generating Instructions in Virtual Environments (GIVE)

The GIVE task (Generating Instructions in Virtual Environments) involves two participants, one instruction giver and one instruction follower, who engage in a ‘treasure hunt’ through a set of virtual worlds. The task can be won by finding and unlocking a safe and obtaining a trophy from it. It can be lost by stepping onto one of a number of red tiles and activating an alarm. To solve the task, the instruction giver has to guide the instruction follower in navigating through a world, and pressing a particular sequence of buttons. The sequence of buttons corresponds to a code that will, if pressed in the correct order, unlock the safe and release the trophy. There are also a number of distractor buttons present, though, which either have no effect or trigger an alarm. In the original GIVE task [Byron et al., 2009; Koller et al., 2010], the role of the instruction giver is taken by an NLG system of the kind that we will develop in the remainder of this thesis. The NLG system’s action set includes navigation instructions, such as moving to the left/right, going straight, or leaving the room. The system also generates referring expressions, which need to be accurate in order to distinguish intended referents from their distractors. To do this, the virtual worlds also contain a set of landmarks, such as plants or furniture, which can be used as points of reference. The instruction follower, or user, is restricted to a number of non-verbal actions. They can either move to the front, left, right or back, or press a button. They can in addition ask for help by pressing a *help* button or cancel the game by pressing *escape*. Note that even though the user’s actions are confined to non-verbal behaviour, the task still resembles a dialogue setting in that the user is able to react to any instruction that the system produces.

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

3.4.2 The GIVE-2 Corpus

The GIVE-2 corpus [Gargett et al., 2010] is a collection of (63 English and 45 German) human-human dialogues on the GIVE task that was collected in a Wizard-of-Oz study to shed light on the strategies that human instruction givers employ when giving navigation instructions and referring expressions to their interlocutors. Participants in this scenario played three games in three different virtual worlds. After the first game, they switched the roles of instruction giver and instruction follower. Interestingly, human instruction givers employ a range of fundamentally different instruction giving strategies as exemplified by the three instruction sequences in Figure 3.8 (which all refer to the same situation). Instructions differ in their length, in their level of abstraction and in their semantic choices. Three main instruction types can be distinguished.

- The first instruction sequence guides the user by a *high-level* navigation strategy. It makes explicit reference to the dialogue history and to locations that the instruction follower has visited previously and is expected to remember (including how to get there). The high-level navigation strategy also makes use of the structure of the environment by referring to doors, paths and rooms.
- The second instruction sequence, in contrast, relies exclusively on guiding the instruction follower by *low-level* navigation. Every required action is explicitly verbalised and there is no abstraction or reference to the environmental structure or dialogue history. We can say that *high-level* navigation instructions represent contractions of *low-level* navigation instructions.
- The third instruction sequence, finally, lies in between the two extremes. While it takes advantage of the environmental structure and visual information, there are no references to the dialogue history. We call this mode of instruction giving *mixed*.

3.4.3 Corpus Annotation

In order to facilitate the automatic analysis of the GIVE corpus dialogues and to provide our learning agent with information about the target domain, we anno-

3. A Joint Learning Agent for Situated Interaction

High-level	
ID	Instruction
1.a	now back to the room with the lamp
1.b	except when you get in take the door to your right
1.c	go to the end and to the left
1.d	the green button next to couch
1.e	<i>[presses green button]</i>

Low-level	
ID	Instruction
2.a	now move forward
2.b	and go through the door on your right
2.c	press the red
2.d	next to the blue
2.e	<i>[presses red button]</i>
2.f	now go forward
2.g	and to the left
2.h	go through press the green
2.i	next to couch <i>[presses green button]</i>

Mixed	
ID	Instruction
3.a	now go through the door
3.b	through the opening
3.c	go through the opening
3.d	to the right of the lamp keep going through the room
3.e	and make a left click that green button
3.f	in front of you <i>[presses green button]</i>

Figure 3.8: Examples of instructions which can be categorised as high-level, low-level and mixed instruction mode (all describing the same situation) taken from the GIVE corpus. The arrows in the maps on the left show the route segment that is described in each instruction. The instruction follower's initial position is indicated by the person in the lower-left room.

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

Table 3.1: Annotation scheme for the GIVE corpus grouped by feature types.

Type	ID	Feature	Values
Utterance	f_1	utterance_type	confirm, repair, RE, navigation
Environment	f_2	discriminating_colour_referent	true, false
	f_3	discriminating_colour_distractor	true, false
	f_4	number_of_distractors	1, 2, 3 or more
	f_5	number_of_landmarks	1, 2, 3 or more
	f_6	is_visible_and_near_object	true, false
Navigation	f_7	navigation_level	high_level, low_level
	f_8	navigation_content	destination, orientation path, ‘straight’, direction
RE	f_9	referent_colour_mentioned	true, false
	f_{10}	distractor_colour_mentioned	true, false
	f_{11}	distractor_mentioned	true, false
	f_{12}	landmark_mentioned	true, false
	f_{13}	spatial_relation	left, right, above, below, next_to
User	f_{14}	within_dialogue_history	true, false
	f_{15}	user_reaction	perform_desired_action, perform undesired_action, request_help, hesitate

tated the English set of dialogues according to the annotation schema shown in Table 3.1. The annotations concern five areas: (1) the utterance itself (its type), (2) the spatial environment, i.e. the situational setting in which an instruction is produced, (3) the semantic choices of a navigation instruction, (4) the semantic choices of a referring expression, and (5) the user’s reaction to an instruction. The user reaction feature is key. We will use it later to provide feedback to the agent’s learning process. Interactions were annotated by two independent annotators. To determine their agreement, Cohen’s kappa score was computed according to

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}, \quad (3.21)$$

3. A Joint Learning Agent for Situated Interaction

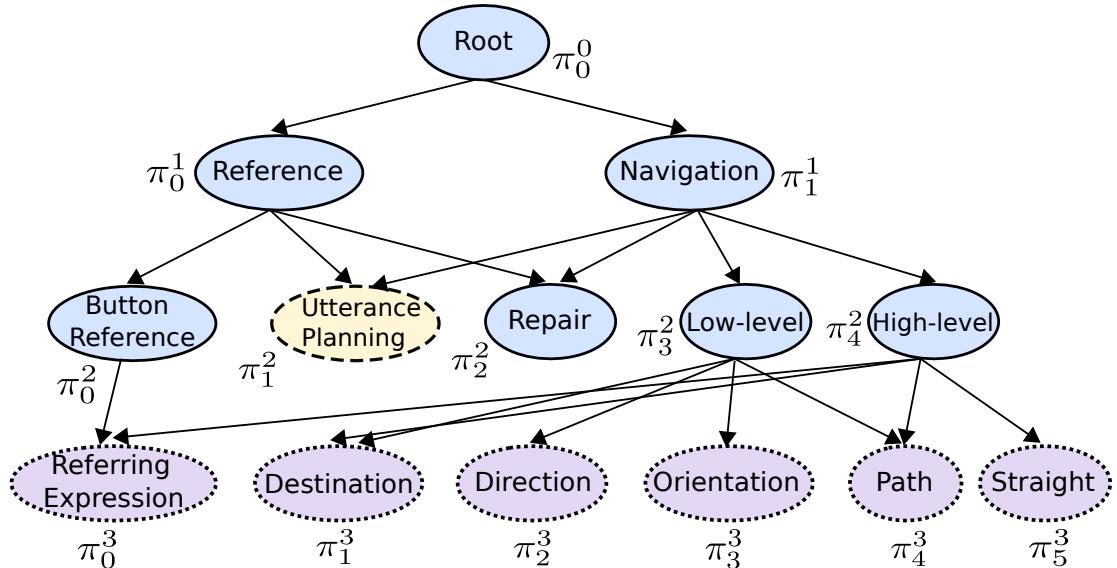


Figure 3.9: Hierarchy of learning agents for content selection (solid lines), utterance planning (dashed lines) and surface realisation (dotted lines) of navigation and referring expression generation in situated scenarios. The arrows indicate the flow of control as it is passed down from parents to child agents. The agents are indexed by their policies $\pi_0^0 \dots \pi_5^3$ for SMDPs $M_0^0 \dots M_5^3$.

where $\text{Pr}(a)$ refers to the observed agreement between the annotators and $\text{Pr}(e)$ refers to the agreement that could have occurred by chance. A kappa score of 1 indicates complete agreement, whereas a score of 0 indicates complete disagreement. For our annotations, the computed kappa score is 0.78, indicating a high inter-annotator agreement. This is necessary to ensure that the annotated categories are reliable, especially given that semantic and pragmatic categories can be of a very subjective nature. See [Carletta \[1996\]](#) for details.

In the following, we will use the annotations to design a hierarchy of learning agents and to design a simulated learning environment for the agent. They are available from http://www.macs.hw.ac.uk/~nsd1/Research_files/annotations.zip.

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

3.4.4 Hierarchy of Learning Agents

As a more concrete description of how knowledge and actions are passed between agents, Figure 3.9 shows the hierarchy of learning agents we designed for the GIVE task. It comprises 14 different agents whose policies can be roughly categorised as tasks of content selection ($\pi_0^0, \pi_0^1, \pi_1^1, \pi_0^2, \pi_2^2, \pi_3^2$ and π_4^2), utterance planning (π_1^2) and surface realisation ($\pi_{0...5}^3$). Note that information is always passed between learning agents in the form of state updates that follow user or system actions.

Content selection is responsible for all semantic decisions made by the learning agent, such as whether to choose a high- or low-level navigation strategy, whether to mention a referent's colour or not, etc. **Utterance planning** focuses on how to organise semantic content into a distinct set of messages. For example, should a set of instructions be aggregated or presented separately, what thematic structure should be used, etc. **Surface realisation** finally chooses a realisation for the utterance from a set of candidates for our six instruction types. For a *joint optimisation*, these 14 agents would share certain knowledge variables among them. This shared knowledge is pre-defined by the system designer and gives us the opportunity to optimise subtasks jointly rather than in isolation. It allows the learning agents to consider different types of decisions interdependently that affect the trade-off between *detail* and *efficiency* in situated interaction. At the same time, it preserves the benefits of a modular architecture.

Generation always begins with the root agent M_0^0 (indexed by its policy π_0^0) which has the option of taking primitive actions or invoke composite actions of reference or navigation. In the latter case control is passed to a child subtask, agent M_0^1 for reference or agent M_1^1 for navigation, respectively. The flow of control is indicated by the arrows in Figure 3.9. During the process of generating an utterance, control is passed between agents, such as from parent to child when a subtask is called, and from child back to parent once a subtask has terminated. Whenever control is transferred back to the root agent, an episode has been completed and execution terminates. One episode (from state s_0 to state s_T) corresponds to one utterance.

The complete state-action space of the hierarchical learning agent has a size

3. Experimental Setting

of $\sum_{i,j} |S_j^i \times A_j^i| = \sum_{i,j} [(\prod_{k(i,j)} |f_{k(i,j)}|) \times |A_j^i|] = 1,480,869$.¹ Here, (i, j) represents an agent in the hierarchy, $f_{k(i,j)}$ represents the feature set of agent (i, j) and k refers to features k in agent (i, j) . In contrast, a flat agent using the same states and actions would have the (very large) state-action space of $|S \times A| = (\prod_k |f_k|) \times |A| = 3 \times 10^{57}$, indicating the advantage of using a hierarchical decomposition for more scalable decision making. The complete state-action space of the hierarchical agent (and the pre-specified shared knowledge variables for a *joint optimisation*) are given in Appendix A.

3.5 Experimental Setting

In this section, we will design a simulated environment from the annotated corpus data and compute a reward function based on a human evaluation study to assist our agent in learning an optimal NLG policy for situated interaction.

3.5.1 The Simulated Environment

Typically, a reinforcement learning agent needs to be exposed to a large number of interactions during training to learn an optimal policy. Since it is impractical to use real users for these interactions, we use a simulated environment instead and estimate it from our annotations of the GIVE corpus. Our goal is to simulate different spatial surroundings in which the agent can try a multitude of action strategies in order to learn an optimal one by trial and error. The effect of each action will be simulated in the form of a user reaction from among $Y = \{\text{perform desired action}, \text{perform undesired action}, \text{wait}, \text{request help}\}$. We assume that users are cooperative so that a good action strategy from the system always

¹ The detailed calculation involves computing the sum over all possible state-action pairs per agent. For the agents specified in Appendix A, this is $(2 \times 2 \times 5 \times 5 \times 3 \times 5) + (2 \times 2 \times 3 \times 5 \times 2 \times 2 \times 2 \times 2 \times 5) + (2 \times 2 \times 3 \times 5 \times 2 \times 2 \times 2 \times 2 \times 2 \times 14) + (2 \times 2 \times 2 \times 4 \times 4 \times 5 \times 9 \times 3 \times 18) + (3 \times 2 \times 2 \times 3 \times 2 \times 5 \times 2 \times 2 \times 2 \times 3 \times 2 \times 5) + (2 \times 3 \times 2 \times 2 \times 2 \times 3 \times 2 \times 3 \times 6) + (3 \times 2 \times 2 \times 2 \times 4 \times 2 \times 2 \times 4) + (4 \times 7 \times 4 \times 8 \times 3 \times 19) + (3 \times 5 \times 4 \times 7 \times 3 \times 15) + (5 \times 4 \times 3 \times 3 \times 9) + (5 \times 7 \times 7 \times 7 \times 3 \times 23) + (6 \times 5 \times 5 \times 3 \times 13) + (2 \times 2 \times 3 \times 4 \times 2 \times 5 \times 2 \times 2 \times 3 \times 10) + (2 \times 2 \times 2 \times 3 \times 3)$, in the order in which the agents appear in Appendix A. In the non-hierarchical RL case, numbers have to be multiplied instead of summed, because no hierarchical decomposition applies.

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

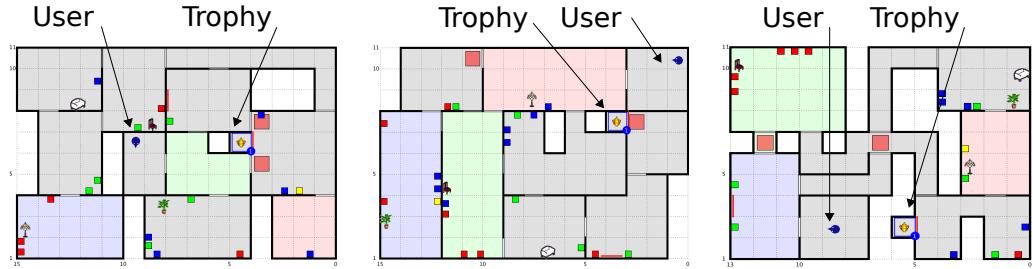


Figure 3.10: Illustration of the training worlds that are the basis of the simulated environment. All worlds require skills for navigation and disambiguation at a medium level of difficulty.

results in the user performing the desired action. All other user reactions indicate a non-optimal system action.

Our simulated environment is based on two Naive Bayes classifiers, one for simulating user reactions (the classes) to referring expressions and one for simulating user reactions to navigation instructions. We use two separate classifiers rather than one because different feature sets are relevant to each system action type. For simulating user reactions to referring expressions, we use the following features $X = \{x_i\}$:

- **discriminating_colour_referent** $x_0 = \{\text{true}, \text{false}\}$, indicates whether the referent's colour is uniquely identifying or not,
- **discriminating_colour_distractor** $x_1 = \{\text{true}, \text{false}\}$, indicates whether any of the distractor's colours are uniquely identifying or not,
- **number_of_distractors** $x_2 = \{0, 1, 2, 3, \text{more}\}$, indicates the number of distractors present, if any,
- **number_of_landmarks** $x_3 = \{0, 1, 2, 3, \text{more}\}$, indicates the number of landmarks present, if any,
- **is_visible_and_near** $x_4 = \{\text{true}, \text{false}\}$, indicates whether the referent button is near and visible to the user (the conditions to press a button),
- **referent_colour_mentioned** $x_5 = \{\text{true}, \text{false}\}$, indicates whether the system's instruction included the colour of the referent, and
- **within_dialogue_history** $x_6 = \{\text{true}, \text{false}\}$, indicates whether the button is already in the dialogue history, e.g. because it has been pressed before.

3. Experimental Setting

For simulating user reactions to navigation instructions, we use the following features $Z = \{z_i\}$:

- **number_of_landmarks** $z_0 = \{0, 1, 2, 3, \text{ more}\}$, indicates the number of landmarks present, if any,
- **is_visible_and_near** $z_1 = \{\text{true, false}\}$, indicates whether the button is visible and near (or whether we need to navigate further towards it),
- **navigation_level** $z_2 = \{\text{high-level, low-level}\}$, indicates whether the system’s instruction was a high- or low-level type instruction,
- **navigation_content** $z_3 = \{\text{destination, direction, orientation, path, straight}\}$, indicates the type of navigation instruction generated, and
- **within_dialogue_history** $z_4 = \{\text{true, false}\}$, indicates whether the next target (a button, room or other object) is already in the dialogue history.

Using these feature sets, we predict user reactions from our annotations of the GIVE corpus by sampling from the distribution $P(Y|X)$ for referring expressions, and by sampling from $P(Y|Z)$ for navigation instructions.

All features describing the environment, such as the number of buttons or landmarks present, were simulated from unigram language models estimated from the GIVE corpus. These features were simulated with the same distribution as they occur in the GIVE corpus, but deliberately so that the agent would encounter as many different settings as possible, i.e. not being restricted to the GIVE worlds¹ shown in Figure 3.10.

To train our classifiers, we used the Weka toolkit Witten and Frank [2005]², and evaluated them in a ten-fold cross validation experiment. For referring expressions, our classifier achieved an accuracy of 78% and for navigation instructions an accuracy of 86%, yielding an average of 82%. As a baseline, a ZeroR (majority class) classifier yields an average accuracy of 69% by always voting for the most likely option.

Training from a simulated environment necessarily means that the agent’s learnt policy depends entirely on the quality of the simulation and its underlying

¹The worlds of the GIVE corpus, which informed our training data, can be downloaded from <http://www.give-challenge.org/research/page.php?id=software> [accessed January 25, 2013].

²www.cs.waikato.ac.nz/ml/weka/ [accessed January 25, 2013]

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

data set. It is therefore important to determine the accuracy of the classifier against a baseline. Nevertheless, even if the classifier reaches a high accuracy on test data, there is no guarantee that the agent’s policy will perform well if users encountered in real interactions diverge significantly from the users encountered during training. Recent research has therefore explored online learning, i.e. to learn from real users’ behaviour during an ongoing interaction [Cuayáhuitl and Dethlefs, 2011b; Gašić et al., 2011].

3.5.2 A Data-Driven Reward Function

According to the PARADISE framework [Walker et al., 1997, 2000], the performance of a (spoken) dialogue system can be modelled as a weighted function of task success and dialogue cost measures (e.g., number of turns, interaction time). We argue that PARADISE is also useful to assess the performance of an NLG system that performs the tasks of content selection, utterance planning and surface realisation, since both objective measures (e.g., task success) and subjective measures (e.g., ease of understanding) seem to be equally relevant for NLG systems. To identify the strongest predictors of user satisfaction (US) in situated dialogue/NLG systems, we will perform an analysis of subjective and objective dialogue metrics based on PARADISE in this section, and then use the resulting performance function as a reward function to train our learning agent in a situated environment.

3.5.2.1 A Wayfinding Study: Experimental Setting

Our aim will be to compute a performance function that has general applicability for dialogue or NLG systems that are deployed in a situated domain. Our main argument for the transferability of one performance function to different systems will be its underlying graded task success metric that takes into account task difficulty. In addition, Walker et al. [2000] provide evidence for the claim that systems of similar domains (in their case information-seeking systems) yield similar performance functions.

We will in the following base our analysis on data collected with a text-based indoor wayfinding system, reported in Cuayáhuitl et al. [2010] with a detailed

3. Hierarchical Reinforcement Learning for NLG

evaluation reported in [Dethlefs et al. \[2010\]](#). The data was collected from 26 participants with an age average of 22.5 and a gender distribution of 16 female (62%) and 10 male (38%). Each participant received six navigation tasks which involved interacting with the system in natural language to obtain route instructions to a pre-specified location in a complex university building. Subsequently, participants had to walk to the target location by following the instructions as closely as possible and without asking for further help along the way. An assistant followed them to take note of their task success and confusions. Participants could give up at any time after having tried without success. On the whole, we collected 156 dialogues. Tasks were executed pseudorandomly (from a uniform distribution) and varied in length so that neither the order of task execution nor their length would have an impact on the results. Users rated the system according to a set of subjective measures after each navigation task on a 1-5 Likert scale (where 5 represents the best rating). The combination of these subjective measures indicate the overall user satisfaction with the system which we will aim to predict from a set of objective measures.

3.5.2.2 Graded Task Success

Typically, the task success of systems is measured using a simple binary success/failure metric:

$$\text{BTS} = \begin{cases} 1 & \text{for Finding the Target Location (FTL),} \\ & \quad \text{with or without problems} \\ 0 & \text{otherwise.} \end{cases}$$

One problem with this metric however is that it does not take into account whether the user encountered any problems or confusions before being successful. Users, on the other hand, do perceive such differences and express them in their ratings of the system (as we will see later in this section). We therefore suggest to use a graded task success metric [\[Tullis and Albert, 2008\]](#) instead which penalises

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

Table 3.2: Mean values of subjective measure results (adapted from [Walker et al. \[2000\]](#)) based on 156 dialogues (standard deviations of values are given behind the \pm sign).

Measure	Description	Score
(Q1) Easy to Understand	Was the system easy to understand?	4.46 ± 0.8
(Q2) System Understood	Did the system understand what you asked?	4.65 ± 0.8
(Q3) Task Easy	Was it easy to find the location you wanted?	4.29 ± 0.9
(Q4) Interaction Pace	Was the pace of the interaction appropriate?	4.63 ± 0.5
(Q5) What to Say	Did you always know what you could write?	4.66 ± 0.7
(Q6) System Response	Did the system reply to you promptly?	4.56 ± 0.6
(Q7) Expected Behaviour	Did the system work the way you expected?	4.45 ± 0.8
(Q8) Future Use	Would you use the system in the future?	4.31 ± 0.9
User Satisfaction (%)	How good was the system overall?	90.0 ± 7.3

task difficulty in wayfinding in several ways¹.

$$GTS = \begin{cases} 1 & \text{for FTL without problems} \\ 2/3 & \text{for FTL with small problems} \\ 1/3 & \text{for FTL with severe problems} \\ 0 & \text{otherwise.} \end{cases}$$

The value of 1 was assigned when users found their target location without any problems. The value of 2/3 was assigned when users found the target location with slight confusions (such as getting lost once), the value of 1/3 was assigned when users had severe problems (such as getting lost multiple times) before finding the target location, and the value of 0 was assigned when users were unsuccessful.

3.5.2.3 Objective and Subjective Measures

Table 3.2 shows all subjective rating categories together with their description, mean values and standard deviations. They indicate an overall user satisfaction of 90%. Objective measures of the dialogues were collected automatically and are presented in Table 3.3 together with descriptions, results and standard deviations.

¹In [Dethlefs et al. \[2010\]](#), we compare different ways of penalising task difficulty by assigning a range of different values. Here we focus on the metric has achieved the best results in the earlier study.

3. Hierarchical Reinforcement Learning for NLG

Table 3.3: Mean values of objective measure results based on 156 dialogues, organised in three groups: dialogue efficiency, dialogue quality, and task success. Standard deviations are given behind the \pm sign.

Measure	Description	Score
Dialogue Efficiency		
System Turns	How many system turns per interaction?	2.30 \pm 0.3
User Turns	How many user turns per interaction?	1.52 \pm 0.5
System Words per Turn	How many system words per turn?	41.30 \pm 4.0
User Words per Turn	How many user words per turn?	4.79 \pm 2.1
Interaction Time (secs.)	How long was the interaction in seconds?	22.14 \pm 18.4
Session Duration (secs.)	How long was the session in seconds?	2014.62 \pm 393.2
Dialogue Quality		
Parsed Sentences (%)	How many user utterances were parsed?	16.7 \pm 16.0
Spotted Keywords (%)	How many needed keyword spotting?	79.9 \pm 17.0
Unparsed Sentences (%)	How many remained unparsed?	3.4 \pm 0.5
Task Success		
Binary Task Success (%)	How many successful interaction?	94.9 \pm 8.3
Graded Task Success (%)	How many interactions without problems?	87.6 \pm 8.3

We can see that in terms of dialogue efficiency metrics, interactions were generally short. Once users had received a set of instructions, they tended to not ask further. With regard to dialogue quality, we can see that most user input was analysed using a keyword spotter. Finally, in terms of task success, we observe a high binary task success, but a lower graded task success.

To predict user satisfaction in a situated NLG system, the objective measures concerning dialogue efficiency and task success are relevant. Dialogue quality measures are neglectable for an NLG system. In addition, the subjective metrics will play a vital role because we will aim to optimise NLG decision making so as to maximise (predicted) subjective user ratings.

3.5.2.4 Correlation Analysis

In order to test the hypothesis that graded task success metrics are more informative with regard to user satisfaction than binary metrics, we performed a correlation analysis on the data. Table 3.4 summarises the results. First, we can see that while the binary metric correlated moderately with user satisfaction (.43),

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

Table 3.4: Correlation coefficients between task success and user satisfaction measures (significant at $p < 0.05$), where *n.s.* means not significant’.

Metric	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Overall
BTS	.47	.20	.53	.21	.20	<i>n.s.</i>	.31	<i>n.s.</i>	.43
GTS	.49	.19	.71	.24	.18	<i>n.s.</i>	.40	.40	.55

the graded metric correlates higher (.55). In particular, the binary metric shows lower correlations according to subjective measures Q1 (‘Easy to Understand’), Q3 (‘Task Easy’), Q4 (‘Interaction Pace’) and Q7 (‘Expected Behaviour’). The difference is particularly obvious with respect to Q3, where the binary metric shows a moderate correlation, but the graded metric shows a high correlation. Finally, while the binary metric shows no correlation with Q8 (‘Future Use’), the graded metrics shows a moderate correlation. These results are significant at $p < 0.05$, and confirm our hypothesis that graded task success metrics are more informative with regard to user satisfaction than binary metrics. Let us now compute the most reliable predictors of user satisfaction.

3.5.2.5 Estimating a Performance Function

In order to identify the relative contribution that different factors have on the variance found in user satisfaction scores, we performed a standard multiple linear regression analysis on our data. We normalised all task success and cost values (dialogue efficiency and dialogue quality, c.f. Table 3.3) to account for the fact that they can be measured on different scales (seconds, percentages, sum, etc.), according to

$$\mathcal{N}(x) = \frac{x - \bar{x}}{\sigma_x}, \quad (3.22)$$

where σ_x corresponds to the standard deviation of x and \bar{x} corresponds to the mean of the observed values. We then performed several regression analyses involving these data. Results revealed that the metrics ‘user turns’ and ‘graded task success’ were the only predictors of user satisfaction at $p < 0.05$. The binary task success metric was not significant ($p < 0.39$). This supports our hypothesis that graded task success metrics are more reliable predictors of user satisfaction than binary metrics. Participants are clearly sensitive to problems and confusions

3. Hierarchical Reinforcement Learning for NLG

they encounter during their tasks, and they express them in their system ratings. The following equation [Walker et al., 1997] can be used to compute a performance function for situated (dialogue and NLG) systems:

$$\text{Performance} = (\alpha * \mathcal{N}(k)) - \sum_{i=1}^n \omega_i * \mathcal{N}(c_i), \quad (3.23)$$

where α is a weight on the task success metric k (to be replaced by our task success metrics), and ω_i is a weight on the cost functions c_i . \mathcal{N} represents the normalised value of c_i . Based on the results of our first regression analysis, we ran a second analysis using only those variables that were significant predictors in the first regression, i.e. graded task success and the number of user turns (which are negatively correlated). We obtain the following performance function:

$$\text{Performance} = 0.38\mathcal{N}(GTS) - 0.87\mathcal{N}(UT), \quad (3.24)$$

where 0.38 is a weight on the normalised value of GTS and 0.87 is a weight on the normalised value of UT (the number of user turns). This result is significant at $p < 0.01$ and accounts for 62% of the variation found in US (user satisfaction). Using this reward function (and -1 for each other action), our learning agent is rewarded for short interactions (few user turns) at maximal (graded) task success. User turns correspond to the behaviour with which a user reacts to an utterance (estimated from the simulated environment presented in the previous section). If the user reacts positively (carries out the instructions), task success is rated with 1; if they hesitate once, it is 2/3, if they hesitate more than once it is 1/3 and if they get lost (carry out a wrong instruction), it is 0. In this way the agent receives the highest rewards for the shortest possible utterance followed by a positive user reaction. This reward function is used by all agents $M_0^0 \dots M_4^2$ dealing with content selection and utterance planning. Rewards are assigned after each system instruction presented to the user and the user's reaction (i.e. whenever an agent of $M_0^0 \dots M_4^2$ has reached its goal state). This reward is propagated back to all agents that contributed to the sequence of decisions leading to the instruction.

Using this reward function, we will train our learning agent to jointly optimise content selection and utterance planning decision making so as to achieve

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

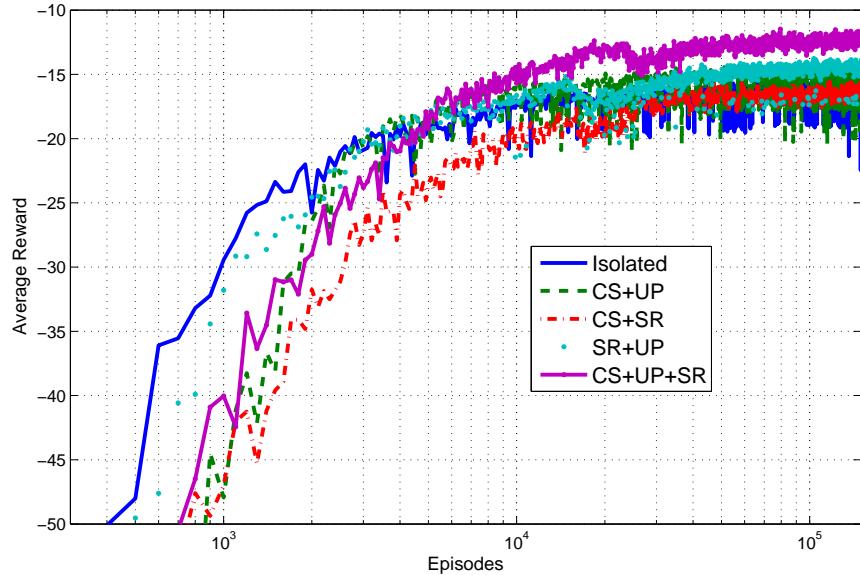


Figure 3.11: Performance (in terms of average reward) of jointly learnt policies in comparison with policies learnt in isolation and several intermediate variants.

maximally successful and short interactions. Surface realisation decision making will for now aim to produce grammatical surface forms.

3.5.3 Training Parameters

We trained all policies for 150 thousand episodes using the following parameters. The step-size parameter α (one per agent), which indicates the learning rate, was initiated with 1 and then reduced over time by $\alpha = \frac{1}{1+t}$, where t is the time step. The discount rate γ , which indicates the relevance of future rewards in relation to immediate rewards, was set to 0.99, and the probability of a random action ϵ was 0.01. See Sutton and Barto [1998] for details on these parameters or Sections 3.2 and 3.3.

3.6 Experimental Results

Figure 3.11 compares the average rewards (averaged over ten runs) of the following set of policies: (a) fully optimised (CS+UP+SR), (b) jointly optimised content selection and utterance planning (CS+UP), (c) jointly optimised content selection and surface realisation (CS+SR), (d) jointly optimised surface realisation and utterance planning (SR+UP), and (e) isolated optimisation (Isolated). We can see that joint optimisation of all three behaviours achieves the highest overall rewards over time and the fully isolated optimisation obtains least rewards. An absolute comparison of the average rewards (rescaled from 0 to 1) of the last 1000 training episodes of each policy shows that the fully joint behaviour improves the fully isolated behaviour by 27%. It improves content selection and utterance planning by 19%, content selection and surface realisation by 23%, and surface realisation and utterance planning by 13%. These differences are all significant at $p < 0.0001$ and their effect sizes range from $r = 0.86$ to $r = 0.98$. In addition, the joint behaviour of content selection and utterance planning improves the isolated baseline by 9%, content selection and surface realisation improves isolated behaviour by 5%, and surface realisation and utterance planning improves isolated behaviour by 16%. The differences are significant at $p < 0.05$.

With regard to content selection, the jointly optimised policy learns to prefer high-level navigation over low-level navigation, but switch the navigation strategy when the user gets confused. It learns to uniquely identify a referent button by preferring the use of a discriminating colour, and otherwise (if neither the referent nor a distractor has a discriminating colour) use either a spatial relation, a distractor or a landmark (in this order of preference). Concerning utterance planning, the agent learns to use composite presentations for at most two instructions (and aggregate them) and incremental displays otherwise. It learns to use temporal markers for more than three instructions and place them in thematic position. In terms of surface realisation, grammatical surface forms are learnt.

As an example of the jointly learnt behaviour, please see Table 3.5. The agent starts using a high-level navigation strategy for efficient instruction giving. As the user hesitates, though, and shows signs of confusion, the agent switches its navigation strategy to low-level and rephrases the earlier instruction with an

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

Conversant	Utterance
USR	<i>starts game: ‘How do I get to the trophy?’</i>
SYS	Move to the end of the hall and then hang left.
USR	<i>hesitates</i>
SYS	Turn 90 degrees right, then proceed straight.
USR	<i>executes instructions</i>
SYS	Now move until the end of the corridor and bear left.
USR	<i>executes instructions</i>
SYS	Very good. Now proceed to the red button left of the blue.
USR	<i>executes instructions</i>
SYS	Click it.
USR	<i>performs desired action: presses red button</i>
SYS	Very good.

Table 3.5: Sample dialogue using the policy optimised jointly. In terms of content selection, the system starts using a high-level navigation strategy, and then switches to low-level as the user gets confused. When the target route segment is navigated, it switches back to high-level. In terms of utterance planning, the system prefers composite presentations for less than three instructions, and incremental displays otherwise. Surface realisation generates only grammatical surface forms without information from a language model.

incremental presentation strategy. This means that first one part is generated ('Turn 90 degrees right, then proceed straight'), and the second part ('Now move until the end of the corridor and bear left') is only presented to the user when the first set of instructions was executed successfully. Subsequently, the agent switches back to high-level and generates a destination instruction to the next referent ('the red button left of the blue'). As the user arrives there, they receive the instruction to click the button. Again this split between the destination description to the button and the instruction to push it is an example of incremental instruction giving to ease the user's comprehension. Note that surface realisation choices are not ideal in all instructions. The agent's goal during training has been only to learn grammatical surface forms. The agent has not had information about the frequency with which different variants occur in a language model of the domain. We will address this problem in Chapter 5.

3. Experimental Results

Conversant	Utterance
USR	<i>starts game: ‘How do I get to the trophy?’</i>
SYS	Move to the end of the hall and then hang left.
USR	<i>hesitates</i>
SYS	Turn 90 degrees right, proceed straight and walk down the corridor. Then bear left.
USR	<i>turns right, requests help</i>
SYS	Move to the end of the hall and then hang left.
USR	<i>undesired action: turns left</i>
SYS	Wait! Head back, then ahead and then follow the corridor.
USR	<i>executes instructions</i>
SYS	Very good. Now proceed to the red one left of the blue and click the button.
USR	<i>undesired action: presses blue button</i>
SYS	No, not this one! I meant the other button. Choose it.
USR	<i>hesitates</i>
SYS	Hit the red.
USR	<i>performs desired action</i>
SYS	Very good.

Table 3.6: Sample dialogue using the policy optimised in isolation. In terms of content selection and utterance planning, the agent prefers efficient instruction giving. It uses high-level navigation, aggregates utterances and presents them jointly. In terms of repair, only the immediately preceding utterance is repaired. Surface realisation again generates grammatical surface forms.

In contrast to the jointly learnt behaviour, Table 3.6 shows an example dialogue generated with the isolated policy. Again, the agent starts using high-level navigation to encourage an efficient interaction. As the user gets confused, the agent switches its navigation strategy to low-level and rephrases the utterance. However, in contrast to the joint example, only the content selection strategy is changed, not the utterance planning strategy. As in the first instruction ('Move to the end of the hall and then hang left'), a composite presentation strategy is chosen, i.e. all four instructions ('turn 90 degrees right', 'proceed straight', 'walk down the corridor', 'bear left') are displayed in the same utterance imposing a

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

high cognitive load on the user. Consequently, the user is able to only process a part of the utterance (*turns right*) and requests help afterwards. The system now switches back to high-level and generates the very first (unsuccessful) instruction again. This is an example of non-coupled content selection and utterance planning behaviour which resulted from the isolated optimisation. Both components aim for efficient instruction giving. Content selection therefore prefers high-level instructions, because they contract several low-level instructions and lead to shorter interactions (if successful). Similarly, utterance planning aims to generate as few utterances as possible and therefore avoids incremental displays. As a result at this point, the user performs an undesired action (*turns left*) and needs to be stopped by the system ('Wait! Head back, then ahead and then follow the corridor.'). This time the user is able to follow the instructions and the system generates its first instruction to press a button ('the red one left of the blue and click the button'). It again uses a composite presentation which results in a long utterance. Further though, it generates an unfortunate reference chain in the referring expression, which mentions first a substitute 'one', then an ellipsis and only at the end the head noun 'button'. This reference chain is a result of isolated surface realisation and content selection decisions. Since surface realisation in isolation is only concerned with grammaticality, not with likely user confusions, the generated reference chain is considered just as good as one that puts the head noun first. Finally, this referring expression needs to be repaired twice before the user identifies and presses the correct target. These example dialogues also once again underline the importance of graded task success as discussed earlier. While users of both dialogues (joint and isolated) are in the end successful, the user of the jointly optimised dialogue is likely to have a substantially higher user satisfaction than the user interacting with the isolated optimisation system, who would be very frustrated.

3.7 Conclusion

In this chapter we have introduced the core of our approach towards jointly optimised NLG: hierarchical reinforcement learning. We have started the chapter by introducing the reinforcement learning framework first from a theoretical perspec-

3. Conclusion

tive and then as a computational model which is based on the Markov Decision Process. After having considered a flat learning setting, we turned to discuss the hierarchical learning setting, which is based on the Semi-Markov Decision Process and provides the foundation of this thesis. By applying both the MDP and the SMDP model to the example of referring expression generation in a situated domain, we were able to identify one of the most important advantages of hierarchical reinforcement learning: its ability to scale to large state spaces and therefore complex domains due to its *divide-and-conquer* approach.

We then aimed to test the framework’s applicability to real-world systems and to investigate our main hypothesis that joint decision making among the core NLG tasks of content selection, utterance planning and surface realisation would lead to better performance than isolated decision making. To do this, we applied it to a full instruction giving system for situated interaction in the GIVE task. Based on a set of annotations of the GIVE corpus, we were able to design a hierarchical learning agent for the task and estimate a realistic simulated environment of the spatial context and the user to train the agent. We further induced a reward function for situated interaction from data collected in a real-world evaluation study with human users. The main underlying claim here was that task success measures that take into account the difficulties that users encounter in wayfinding would be better predictors of user satisfaction than the often used binary (success/failure) metrics. This claim was confirmed based on correlation and multiple linear regression analyses.

Training and testing the hierarchical learning agent in the simulated environment finally also confirmed our original hypothesis that learning an NLG policy jointly, i.e. with interrelated decision making between the tasks of content selection, utterance planning and surface realisation, would lead to better overall performance than optimising them in isolation. Hierarchical reinforcement learning thus appears to be a promising framework for the development of large and complex natural language generation systems that interact with users in an adaptive way and need to react flexibly to a range of possibly arising problems.

So far we have observed two main limitations of our approach. First, even though the hierarchical setting makes learning a large policy feasible, learning times are still long and we seem to have made insufficient use of prior domain

3. HIERARCHICAL REINFORCEMENT LEARNING FOR NLG

knowledge. Prior knowledge may come in many forms. It may come as general intuitions about a decision making process such as that the same decision does not need to be considered multiple times during a single generation process. Alternatively, we may wish to include prior knowledge on human linguistic preferences we have observed within the domain, such as that humans have a strong preference for mentioning the colour of a referent, even if it does not help its unique identification. Such preferences may not always have an impact on task success so that the learning agent would learn them. Nevertheless, they are often real and may be worthwhile including in an NLG system. The second limitation so far concerns surface realisation choices. Since our learning agent was only driven by considerations of grammaticality, it made a range of unlikely surface realisation choices. Backing the agent up with a language model of the domain may help to mitigate this problem and produce more natural and comprehensible language.

We will address both problems in the following chapters. Chapter 4 will introduce the idea of a hierarchical information state, which allows the inclusion of prior knowledge into the agent’s learning process. Chapter 5 will then combine hierarchical reinforcement learning with graphical models that represent the agent’s generation space. In this way, we will demonstrate how surface realisation decisions can be jointly optimised with other tasks, and at the same time be based on a human language model of the domain.

Chapter 4

A Hierarchical Information State for Constrained Learning

4.1 Introduction

A learning agent that generates utterances in situated interaction is faced with a multitude of choices concerning the final output presented to the user. Many of these choices deal with content selection, that is *what to say*, which will be the focus of this chapter. Given a corpus of the target domain and a generation situation that needs to be communicated to the user, there is typically more than one way to communicate the desired semantics. As an example of the possible content selection choices that an agent can face consider the following (non-exhaustive) list of possible referring expressions for the situation depicted in Figure 4.1. The intended referent is circled on the picture. The instructions vary along the dimensions of whether they include the referent's colour or type, its distractors, close-by landmarks, the referent's position and the dialogue history. The chosen dimensions are indicated in parentheses for each example.

- Push the red button. (*type, colour*)
- Push the left button. (*type, position*)
- Push the button. (*type*)
- Push red. (*colour*)
- Push the button beside the green. (*type, distractor*)
- Push the button right of the plant. (*type, landmark*)
- Push the second button from the right. (*type, position*)
- Push the second button from the door frame. (*type, landmark*)
- Push the button between the plant and the green button. (*type, landmark, distractor*)
- Push the button beside the button. (*type, distractor*)
- Push the same button as before. (*type, discourse history*)

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

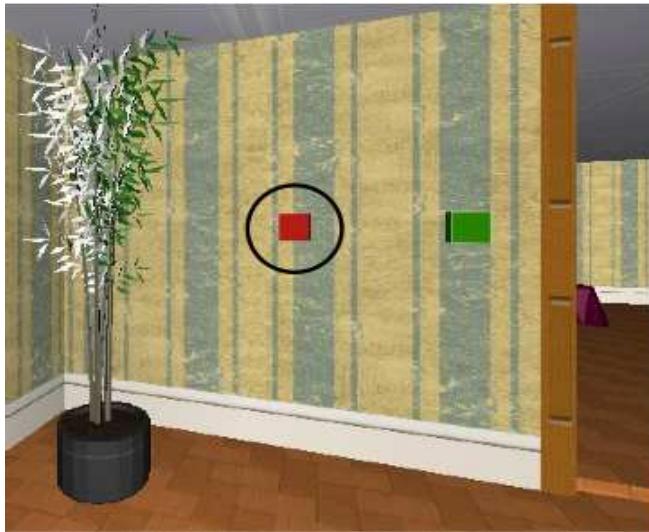


Figure 4.1: Example of a referring expression generation situation in the GIVE world where more than one description of the target referent (circled) is possible.

• ...

An agent that is faced with such a vast amount of choices needs a principled means to decide among them and make the best choices according to the situation and the user. In the previous chapter, we presented a hierarchical reinforcement learning agent that learns to optimise its decision making for situated interaction so as to reach maximal task success and maximal user satisfaction. However, optimising these two measures does not always lead deterministically to a single output. Often we are faced with a set of options which are equally optimal with respect to task success and user satisfaction. The question then arises how we should best choose among them? The agent we developed in Chapter 3 so far chooses among them randomly which begs the further question of whether we can do better than random and how?

Related work on generation in situated domains has focused on supervised learning techniques to address the content selection problem. Stoia et al. [2006] use decision trees to learn content selection rules for noun phrases in a situated generation setting. They evaluate their approach in terms of human ratings and show that roughly two thirds of their generated noun phrases were judged equal

4. Introduction

or better than human noun phrases for the domain. Similarly, [Dale and Viethen \[2009\]](#) and [Viethen et al. \[2011\]](#) use decision trees to learn content selection rules for referring expression generation in spatial settings. Their evaluations are based on accuracy but show that decision trees are only partially useful to predict human referring behaviour. Importantly, they show that human referring expressions are to a large part influenced by the preferences of individual speakers [[Viethen, 2011](#)].

Garoufi and Koller use a slightly different approach for NLG in situated interaction. They use AI planning to make a first set of content selection decisions [[Garoufi and Koller, 2010](#)] and then apply a maximum entropy model to resolve the remaining nondeterminacy among options that are equally good according to the planning component [[Garoufi and Koller, 2011a](#)]. An evaluation of their approach in terms of accuracy, human-likeness and task success looks promising.

All of these approaches demonstrate that supervised learning is an attractive method to learn behaviour from a corpus, discover interdependencies between choices and perform interrelated decision making. A problem that typically arises with supervised learning accounts however is their limited ability to generalise to unseen problems and new situations. Systems exposed to circumstances that differ substantially from their training environment can often find themselves unable to adapt to the new circumstances and produce behaviour that is unpredictable, as observed by [Levin et al. \[2000\]](#). This effect is particularly harmful in domains where decisions are tightly interrelated or where flexible troubleshooting is needed. Both is true of the complex situated interaction task we are addressing in the GIVE domain. A problem related to the system's lack of flexibility in unseen contexts is the need for large amounts of labelled training data and the associated high cost of human annotation. The system needs to be exposed to many examples in order to learn a good and robust behavioural strategy. Finally, in NLG a corpus of the domain cannot be considered a gold standard to the same extent as in other areas of NLP [[Belz and Reiter, 2006; Scott and Moore, 2006](#)]. The fact that a phrase or sentence does not occur in the corpus data does not mean that it is unacceptable or even worse than any phrase or sentence that does occur. It is therefore desirable for any system to learn to make suitable generalisations over its training corpus.

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

Due to the limitations that supervised learning methods face, we argued earlier that hierarchical reinforcement learning is a powerful framework to learn flexible generation policies for complex domains from a limited amount of data. However, a reinforcement learning agent will always just learn the behaviour it receives most rewards for. So far this has been task success and user satisfaction. Supervised learning methods on the other hand are able to discover human preferences in the data even if these do not have an impact on task success or user satisfaction.

Apart from their disability to discover human preferences in learning environments in which this is not the primary source of rewards, reinforcement learning agents can have difficulties in finding an optimal policy in large search spaces and are therefore often limited to small-scale applications. Pruning the search space of a learning agent by including prior knowledge (which can come in the form of human preferences) is therefore attractive, since it allows the agent to find solutions faster, it reduces computational demands, incorporates expert knowledge, and scales to more complex problems than without prior knowledge. Suggestions to use such prior knowledge include [Litman et al. \[2000\]](#) and [Singh et al. \[2002\]](#), who hand-craft rules of prior knowledge obvious to the system designer for dialogue management. Similarly, [Cuayáhuitl \[2009\]](#) suggests using Hierarchical Abstract Machines to partially pre-specify dialogue strategies, and [Heeman \[2007\]](#) uses a combination of RL and an information state to also pre-specify dialogue strategies. [Williams \[2008\]](#) presents an approach of combining Partially-Observable Markov Decision Processes with conventional dialogue systems.

Ultimately, it seems attractive to combine the hierarchical reinforcement learning agent we have designed so far with a rule-based component that captures prior knowledge of our generation domain in the form of human preferences and thereby safely prunes the learning agent’s search space. Such constraints can incidentally also be used to ensure the coherence of the agent’s behaviour, for example to avoid ambiguous references. We will design an agent that makes optimal decisions with regard to task success and user satisfaction and at the same time bases its decisions on human preferences whenever more than one optimal decision is available.

4. Content Selection for Situated Interaction

In the remainder of this chapter, we will first investigate what insights we can gain by applying supervised learning to content selection¹ in the situated interaction scenario. We will analyse what human preferences hold for our domain and how strong they are across individual persons. Our main suggestion will be to systematically pre-specify human preferences in the form of an Information State (IS), which will be applied to our hierarchical NLG scenario resulting in a *Hierarchical Information State* (HIS). The hierarchical IS will subsequently be integrated with our hierarchical reinforcement learning model so as to constrain the learning agent’s decision making whenever strong human preferences for a decision exist. If none exist, or more than one exists, the learning agent will optimise its decision making as usual and learn the best action to take over time. We will then re-train our learning agent and compare the resulting semi-learnt policy with two baselines: the fully-learnt policy developed in the previous chapter and a purely supervised learning-based policy. Results show that the reinforcement learning-based policies outperform the supervised learning-based policy. Furthermore, we will see that the semi-learnt policy produced behaviour is equivalent to the fully-learnt policy, but can be learnt in only a fraction of the time taken by the fully-learnt policy. This suggests that hierarchical reinforcement learning with a hierarchical information state can help to make our decision making model scale to larger search spaces. In this way, larger and more complex problems and domains can be addressed.

4.2 Content Selection for Situated Interaction

We can draw two observations from previous work that has used decision tree-based methods to learn rules for generation in a situated interaction scenario. One strand of work has shown that this direction is promising for inducing generation behaviour [Stoia et al., 2006]. The other strand of work however has pointed out some limitations of this approach, since supervised learning is usually not able to identify the strong individual speaker preferences that can be found in human

¹We focus on content selection in this chapter with the assumption that utterance planning is predominantly guided by considerations of cognitive load and memory not by individual human preferences. Surface realisation will be the detailed topic of Chapter 5.

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

corpora [Dale and Viethen, 2009; Viethen, 2011]. In this section, we will first apply decision trees to learn rules for our own generation domain. We will then analyse whether strong human preferences exist for our domain and how well they are represented by the decision trees.

4.2.1 Decision Trees for Content Selection

In order to learn a set of generation rules for the GIVE task, we use Weka’s [Witten and Frank, 2005] J48 decision tree classifier and apply it to the annotated GIVE corpus introduced in Section 3.4.2 in Chapter 3. In addition, we use a number of spatial features concerning the state of the virtual world. Each attribute in the annotation is used as a target attribute in a separate decision tree and is to be predicted based on the attributes given below for navigation instructions and referring expressions. Importantly, the feature set for each attribute contains the values of all other attributes, so that it is possible to learn context-dependent rules. Specifically, the decision tree classifiers for navigation instructions use the following set of features:

- **corridor** {yes, no},
- **direction change** {yes, no},
- **instruction type** {destination, direction, orientation, path, straight},
- **leave room** {yes, no},
- **navigation level** {high, low},
- **next goal visible** {yes, no},
- **next room known** {yes, no},
- **next room equals** {previous, current, new},
- **number of doors** { ≤ 1 , > 1 },
- **orientation change** {yes, no},
- **previous user reaction** {perform desired action, perform undesired action, request help, wait},
- **repair** {yes, no},
- **straight** {yes, no},
- **user confused** {yes, no},

4. Content Selection for Situated Interaction

- **user reaction** {perform desired action, perform undesired action, hesitate, request help}.

The decision tree classifiers for referring expressions use the following set of features:

- **distractor used** {yes, no},
- **distractor colour discriminating** {yes, no},
- **distractor colour used** {yes, no},
- **distractor near** {yes, no},
- **distractor visible** {yes, no},
- **landmark near** {yes, no},
- **landmark used** {yes, no},
- **horizontal row present** {yes, no},
- **horizontal row used** {yes, no},
- **landmark visible** {yes, no},
- **previous user reaction** {perform desired action, perform undesired action, request help, wait},
- **referent colour discriminating** {yes, no},
- **referent colour used** {yes, no},
- **referent near** {yes, no},
- **referent visible** {yes, no},
- **repair** {yes, no},
- **user confused** {yes, no},
- **user reaction** {perform desired action, perform undesired action, hesitate, request help},
- **vertical row present** {yes, no},
- **vertical row used** {yes, no}.

As an example of the learnt trees, Figure 4.2 shows the decision tree for whether or not the colour of the referent button should be mentioned. On average, the decision trees reached an accuracy of 91% in a 10-fold cross validation. The following sections describe the learnt rules in detail.

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

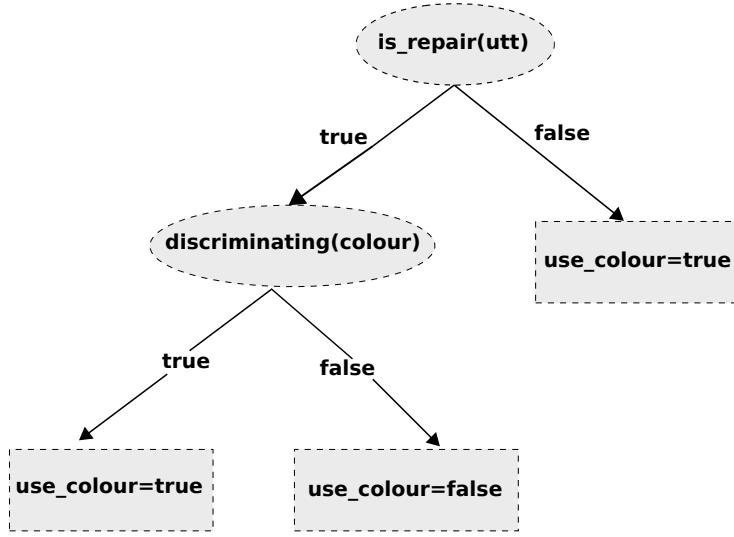


Figure 4.2: Example of a decision tree concerned with whether or not the colour of the referent should be included in a referring expression. It appears that whenever an utterance is not a repair (e.g. a rephrase or repetition) of the previous utterance, the colour should be included.

4.2.1.1 Learnt Rules for Navigation Instructions

The following set of rules was learnt for navigation behaviour. Importantly, the rules prefer a high-level navigation strategy whenever possible.

1. Use high level navigation if any of the following is true:
 - (a) the next room is known.
 - (b) the next goal is visible.
 - (c) the next goal is near.
2. Use an orientation instruction when the user is changing their orientation.
3. Use a direction instruction when the user should change the direction.
4. Use a ‘straight’ instruction when the user should go straight.
5. Use a destination instruction if the next room contains the next goal.
6. Use a path instruction when the user needs to leave the room through a door and there is exactly one door present.
7. Use a path instruction and a direction instruction when the user needs to leave the room through a door and there is more than one door present.

4. Content Selection for Situated Interaction

8. Repair the previous utterance when the user does not react by executing the desired instructions.

4.2.1.2 Learnt Rules for Referring Expressions

The following set of rules was learnt for referring expression generation. It reveals a preference for the use of colour and for using a distractor over a landmark.

1. Include the referent's colour if any of the following is true:
 - (a) the current utterance is not repairing a previous utterance.
 - (b) the current utterance is repairing a previous utterance and the referent's colour is discriminating.
2. Include a distractor if one or more are present, the referent's colour is not discriminating and if any of the following is true:
 - (a) an adjacent distractor has a discriminating colour.
 - (b) there is a distractor in a vertical spatial relation to the referent.
 - (c) there is a distractor in a horizontal spatial relation to the referent.
3. Use the colour of a distractor whenever a distractor is used.
4. Include a landmark if the referent's colour is not discriminating and no suitable distractor is present.
5. Repair the previous utterance when the user does not react by executing the desired reference instruction.

4.2.2 Consistency and Alignment in Human Data

A closer look at the learnt generation rules reveals that there are indeed a number of striking human preferences. With respect to referring expressions the most noticeable tendency is probably the overuse of colour even when colour is not a distinguishing property to uniquely identify a referent. It is represented by rule 1a above which states that colour should always be used except when a previous utterance is repaired. In this case, colour is only used when it is indeed distinguishing. Further tendencies are the preference for identifying a referent with respect to a distractor rather than a landmark (rules 2 and 4), or the preference

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

for mentioning a distractor colour over a spatial relation (rules 2a and 2b and 2c) and a vertical spatial relation over a horizontal relation (rules 2b and 2c).

With regard to navigation instructions, it seems at first sight that humans have a preference for using high level navigation whenever possible (rule 1). In this case however, this is just an apparent preference which results from a generalisation of the supervised learning algorithm. A qualitative data analysis reveals that in fact we are dealing with three different kinds of human instruction givers. One kind strongly prefers the use of high level instructions, another kind prefers the use of low level instructions, and a third group varies between both and uses a mixed navigation strategy. Incidentally, these three kinds of instruction giving are exemplified by the dialogues in Figure 3.8 in Section 3.4.2 in Chapter 3. A crucial observation here is that once instruction givers have chosen a particular strategy, they tend to use it until the end of the dialogue. In fact, it can be observed from the English GIVE corpus data that 75% of all instruction givers use the same navigation strategy in consecutive games, which suggests a strong consistency in instruction giver behaviour. Moreover, in 62.5% of all dialogues (with the same interlocutors), the same navigation strategy is used even over three consecutive games, i.e. after participants have changed roles, indicating a strong inter-speaker alignment. The effect of speakers consistently using one spatial description strategy was also noted by Moratz et al. [2001].

We thus find two prominent kinds of human preferences in the corpus: (1) overall human preferences as observed with respect to referring expressions and (2) individual human preferences as found with regard to navigation instructions. Despite the strength of both preferences, they do not have an obvious impact on task success or user satisfaction in the GIVE corpus dialogues. Nevertheless, the prominence of both tendencies seems to suggest that a learning agent that aims to generate human-like and natural language should take this phenomenon into account. In the next section we will transfer the notion of an information state to NLG and show how it can be used to enforce the observed human preferences.

4.3 Information State

The notion of an information state has traditionally been applied to dialogue management [Bohlin et al., 1999; Bos et al., 2003; Larsson and Traum, 2000; Lemon et al., 2006; Matheson et al., 2000; Quinderé et al., 2007; Ross and Bateman, 2009], where it encodes the current state of a dialogue, i.e. all information currently required to choose the next best dialogue move. Such information can include the context of the dialogue, dialogue participants and their beliefs, the status of grounding or any further piece of information that is relevant for decision making. An information state consists of five components: (1) a set of *informational components* encoding information about the dialogue, (2) *formal representations* of these components, (3) a set of *dialogue moves*, or actions, which lead to updates of the information state, (4) a set of *update rules* which govern the update, and (5) an *update strategy*, which specifies which update rule to apply in case more than one applies at any point in the dialogue. In the following, we will apply the notion of an information state to NLG.

4.3.1 Informational Components

We define the informational components of the information state to represent all linguistic, situational and spatial knowledge required by the learning agent. There are two kinds of knowledge represented as informational components. The first corresponds roughly to the agent’s state space introduced earlier in Section 3.4.4 of Chapter 3 and in Appendix A. All the knowledge available to the agent is also available to the information state. The second kind of informational components is only available to the information state. It encodes the values of actions that the agent has taken. For example, while the agent has only state variables such as $\text{ColourDistractor} = 0$ (when the decision of whether or not to include the colour of a distractor is still pending) or $\text{ColourDistractor} = 1$ (when it has been made), the information state has additional information of the form $\text{ColourDistractorValue} = \text{useColour}$ or $\text{ColourDistractorValue} = \text{dontUseColour}$ indicating the value of the decision. The information state will use these informational components to analyse the

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

state of the generator and provide action constraints whenever appropriate.

4.3.2 Formal Representations

We represent informational components of the information state formally as vectors of integers and strings. Integers are used as in Appendix A, they represent the different values that states of the agent’s knowledge representation can take on. In addition, the information state keeps track of all actions that the agent takes. This information is encoded as strings corresponding to the name of the action taken.

4.3.3 Generation Moves

Generation moves correspond to generation actions exactly as before in Chapter 3, Section 3.4.4 and Appendix A. Just as actions of the learning agent trigger a state transition and an update of the generation state, generation actions trigger an update of the information state to represent the changes caused by the action. Both primitive and composite actions of the learning agent are used as generation moves in the information state.

4.3.4 Update Rules

Update rules are triggered by generation actions and record all changes in the environment that were caused by the action. They roughly correspond to the state transitions in a learning agent (even though in a learning agent they are probabilistic). In our hierarchical learning setting, we will use update rules to specify constraints on the agent’s action selection. For example, we can use update rules to specify human preferences of the kind observed in the previous section. They can be used in two ways. First, we can specify that in a particular state of the world, one action should always be chosen. This principle could apply for strong human preferences such as the overuse of colour observed in the previous section. Second, we can use update rules to exclude certain actions. This is useful in situations where there is a strong human preference *against* an

4. Information State

action. In this case, we can prune the action from the set of all available actions for the state and let the agent choose from the remaining ones.

4.3.5 Update Strategy

The update strategy of an information state decides which action to take next in case more than one action is available. The most simple update strategy would be random selection, but more sophisticated strategies such as ranking can be used. In our scenario, the update strategy will be learnt by the hierarchical reinforcement learning agent. That is, whenever more than one action is available, or no action constraints are specified, the agent learns the best sequence of actions using hierarchical reinforcement learning as in the previous chapter.

4.3.6 Example of an Information State for NLG

To exemplify the notion of an information state for NLG, let us return to the referring expression generation example from before. Assume we have the same set of actions (or generation moves) available: *mention* or *don't mention the referent's colour*, *include* or *don't include a distractor*, *include* or *don't include a landmark*, *include* or *don't include a spatial relation*. Our state representation contains the same information as before: the number of distractors and landmarks, whether or not the referent's colour or position is discriminating and the decisions we have made. We can formalise these features into an information state as follows using the format *informational_component : formal_representation*.

infostate:

```
(referent_colour_discriminating : number,  
number_of_distractors : number,  
number_of_landmarks : number,  
referent_position_discriminating : number,  
colour_mentioned : number,  
distractor_mentioned : number,  
landmark_mentioned : number,  
spatial_relation_mentioned : number)
```

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

Using this information state, we can now specify any number of update rules. Note that for a system that works only based on the information state, an update rule would need to be specified for each action. In our case, this is not necessary, so we only specify those update rules that correspond to the action constraints we wish to apply. For example, we can specify the user preference for using colour as follows.

update_rule:

```
(use_referent.colour  
  precondition: colour.mentioned == 0  
  effect: colour.mentioned ← 1 )
```

This update rule states that whenever the slot of ‘colour_mentioned’ is unfilled, i.e. when the decision of whether to use a colour or not has not been made yet, then ‘use_referent.colour’ is a valid option (but ‘dont_use_referent.colour’ is not when we have not specified a rule for it). It also states that following the action, the informational component ‘colour_mentioned’ should be updated from 0 to 1. In this way, we ensure that the agent will always use a referent’s colour in a referring expression. Since we have not specified any other update rules, the remaining action selection behaviour will be subject to optimisation using hierarchical reinforcement learning as before. This is also the case when more than one action is available at some point: the agent will in this case learn the best sequence of applying them. The exact way in which the information state is integrated into the hierarchical reinforcement learning framework will be the topic of the following sections.

4.4 Combining Hierarchical RL with a Hierarchical Information State

In order to integrate a hierarchical information state into our hierarchical reinforcement learning framework we need to revise three components of the learning model presented in Chapter 3: the definition of an SMDP, the definition of the HSMQ-Learning Algorithm and the definition of the hierarchy of learning agents.

4. Combining Hierarchical RL with a Hierarchical Information State

4.4.1 The Semi-Markov Decision Process Using an Information State

In the previous chapter, we have defined a Semi-Markov Decision Process as a four-tuple model $M_j^i = \langle S_j^i, A_j^i, T_j^i, R_j^i \rangle$, where S_j^i is a set of generation states, A_j^i is a set of generation actions, T_j^i is a probabilistic state transition function specifying the next state s' from the current state s and the action a , and R_j^i is a reward function assigning a numeric reward for each action a that the agent takes in state s . The indices i and j were used to uniquely identify an agent in the hierarchy of learning agents. In order to integrate a hierarchical information state into the learning framework we have defined so far, we need to update the definition of the SMDP model as a five-tuple $M_j^i = \langle S_j^i, A_j^i, T_j^i, R_j^i, I_j^i \rangle$, where S_j^i , A_j^i , T_j^i and R_j^i are as before and the additional element I_j^i is an information state used as knowledge base and rule-based decision maker as exemplified in the previous section. The indices i and j in this case associate an information state uniquely with one of the agents of the hierarchy making the complete information state hierarchical per se. The dynamics of this extended model are shown in Figure 4.3, where action selection is based on a constrained set of actions provided by the information state (IS) update rules. We assume that the names of update rules in I_j^i represent the agent actions A_j^i . The goal of each SMDP is then (as before) to find an optimal policy π_j^{*i} that maximises the reward that the agent receives for each visited state, according to the updated equation

$$\pi_j^{*i}(s) = \arg \max_{a \in A_j^i \cap I_j^i} Q_j^{*i}(s, a), \quad (4.1)$$

where $Q_j^{*i}(s, a)$ specifies the expected cumulative reward for executing constrained action a in state s and then following π_j^{*i} thereafter. In order to learn policies using the notion of constrained actions, we use a modified version of HSMQ-Learning, which is presented in the next section.

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

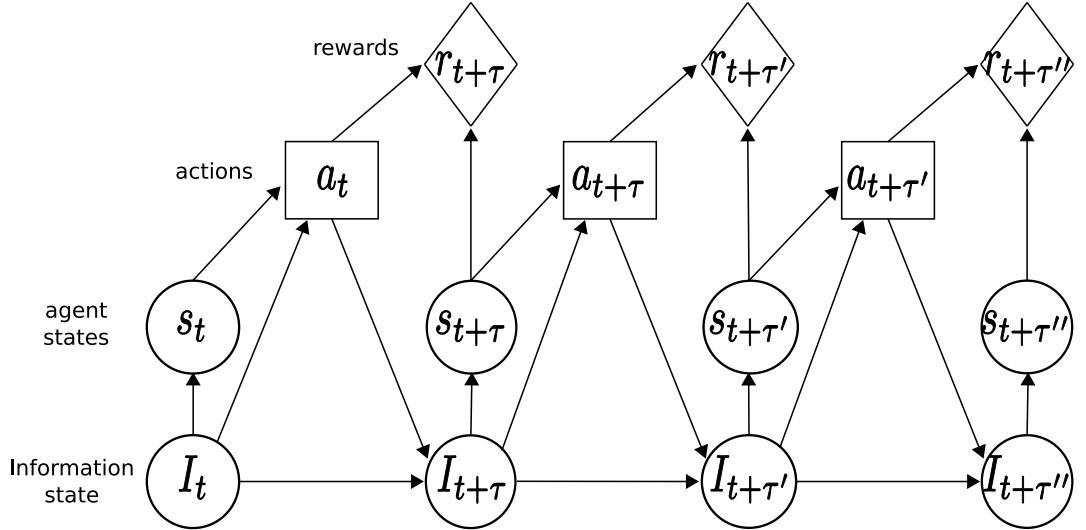


Figure 4.3: A constrained Semi-Markov Decision Process for NLG, where I_t represents the information state at time t , s_t represents a knowledge-compact state, a_t is a generation action (primitive or composite) available in s_t , and r_t is a numerical reward for choosing action a_t . Note that the information state constrains the actions available per state, i.e. action a_t is selected if $a_t \in A \cap I$.

4.4.2 The HSMQ-Learning Algorithm Using Constrained Actions

For learning generation policies using constrained sets of actions to represent human preferences or prior knowledge of a domain, we update the HSMQ-Learning algorithm (Algorithm 2) to a modified version shown in Algorithm 3. This algorithm receives subtask M_j^i and Information State I_j^i used to initialise state s , performs similarly to Q-Learning for primitive actions, but for composite actions it invokes recursively with a child subtask. In contrast to HSMQ-Learning, this algorithm chooses actions from a subset derived by applying the IS update rules to the current state of the world. In this way, the agent can only choose actions from the constrained set of actions rather than the complete set. When the subtask is completed, it returns a cumulative reward $r_{t+\tau}$, and continues its execution until finding a goal state for the root subtask. This process iterates until convergence occurs to optimal context-independent policies, as in HSMQ-Learning. We use

4. Experimental Setting

Algorithm 3 HSMQ-Learning using the Information State approach

```

function IS+HSMQ(InformationState  $I_j^i$ , subtask  $M_j^i$ ) return  $totalReward$ 
     $s \leftarrow$  knowledge-compact state in  $S_j^i$  initialized from  $I_j^i$ 
    3:  $totalReward \leftarrow 0$ ,  $discount \leftarrow 1$ 
    while  $s$  is not a terminal state do
         $\tilde{A} \leftarrow$  Constrained set of actions derived from info state update rules in  $I_j^i$ 
        6: Choose action  $a \in \tilde{A}$  from  $s$  using policy derived from  $Q_j^i$  (e.g.  $\epsilon$ -greedy)
        Execute selected action  $a$  and update the information state  $I_j^i$ 
        if  $a$  is primitive then
            9: Observe one-step reward  $r$ 
            else if  $a$  is composite then
                 $r \leftarrow$  HSMQ(information state of action  $a$ , subtask of action  $a$ )
            12: end if
            Update info state  $I_j^i$  after complete execution of action  $a$ 
             $totalReward \leftarrow totalReward + discount \times r$ 
            15:  $discount \leftarrow discount \times \gamma$ 
            Observe resulting state  $s'$ 
             $Q_j^i(s, a) \leftarrow (1 - \alpha)Q_j^i(s, a) + \alpha [r + discount \times \max_{a' \in A_j^i} Q_j^i(s', a')]$ 
            18:  $s \leftarrow s'$ 
        end while
end function

```

the DIPPER toolkit [Bos et al., 2003] for our implementation of the IS. All action constraints applied to the hierarchy of learning agents are specified in Appendix B.

4.5 Experimental Setting

Before re-training the agent, we will use this section to revisit the reward function and training parameters used during training. The simulated environment will be the same as described in Chapter 3, Section 3.5.1.

4.5.1 A Reward Function for Consistency

In Section 3.5.2 we induced the following reward function from a set of data gathered during a human evaluation study with a situated wayfinding dialogue

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

system. It rewards short interactions at maximal task success.

$$\text{Performance} = 0.38\mathbb{N}(GTS) - 0.87\mathbb{N}(UT). \quad (4.2)$$

We have also assigned a reward of -1 of each action selected in order to prevent the agent from entering into loops. In Section 4.2.2 of this chapter, we have made the additional observation that humans show a strong preference for consistent instruction giving. Humans tend to choose either a low-level, a high-level or a mixed strategy for giving navigation instructions initially and then stick to their chosen strategy until the end of the interaction. To correspond to this tendency, we will inspect for each generated instruction whether its navigation level is consistent with the navigation level of the previous instruction and reward a consistent instruction giving with an additional reward of $+1$.

4.5.2 Training Parameters

As before, we trained all policies for 150 thousand episodes using the parameters specified in Section 3.5.3. The step-size parameter α (one per agent), which indicates the learning rate, was initiated with 1 and then reduced over time by $\alpha = \frac{1}{1+t}$, where t is the time step. The discount rate γ , which indicates the relevance of future rewards in relation to immediate rewards, was set to 0.99, and the probability of a random action ϵ was 0.01. Details on these parameters were discussd in Sections 3.2 and 3.3 or in Sutton and Barto [1998].

4.6 Experimental Results

This section will evaluate the hierarchical reinforcement learning framework with an integrated hierarchical information state from two perspectives: (1) we will observe the effect that the hierarchical IS has on the average cumulative rewards that the learning agent receives over time, and (2) we will compare human ratings for instructions generated using hierarchical reinforcement learning with constrained actions and instructions generated using decision trees alone.

4. Experimental Results

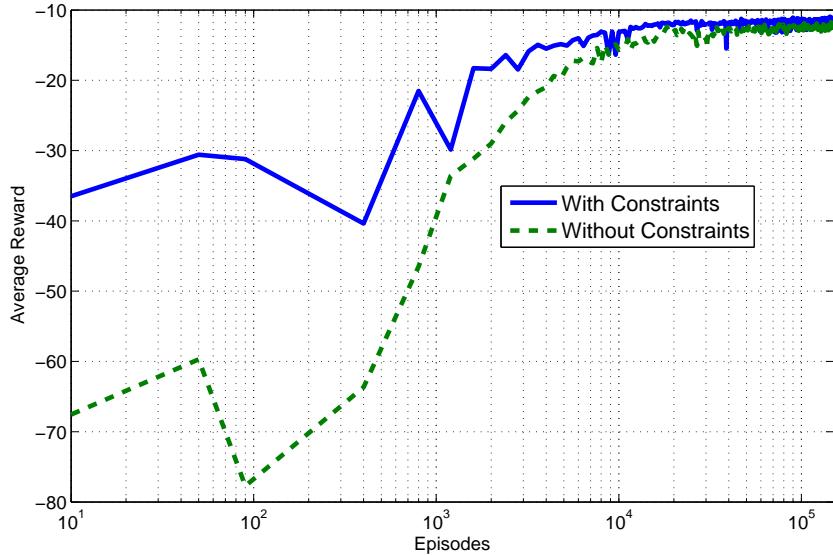


Figure 4.4: Learning curves indicating the rewards (averaged over 10 runs) that the fully-learnt (Without constraints) and the semi-learnt (With constraints) agents received over 150 thousand learning episodes. While both agents learn equivalent policies, the semi-learnt agent learns an optimal policy earlier.

4.6.1 Simulation-Based Results

Figure 4.4 compares the fully-learnt policy from the previous chapter with the semi-learnt policy developed in this chapter with regard to their average reward over time, i.e. the predicted task success and user satisfaction of the simulated interactions. Note that both policies are optimised jointly for content selection, utterance planning and surface realisation. Importantly, we can see that both policies—with and without constraints—eventually learn equivalent behaviours as indicated by the average rewards they receive over time.¹ This provides evidence that specifying constraints on action selection did (in our case) not affect the performance or optimality of the learnt policy. Another important effect we see though is that the semi-learnt policy learns an optimal policy in substantially

¹Note however that the rewards obtained by the constrained learning agent are slightly higher due to the extra reward of +1 that we assign for consistent navigation instructions. This is also reflected in the learning curves in Figure 4.4.

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

less time than the fully-learnt policy. This effect is due to the constraints we specified in the form of update rules in the information state. Since we constrained the agent so as to consider every decision point only once per utterance, learning an optimal sequence of actions for an instruction is accelerated visibly (due to prohibiting unnecessary actions that do not change the state of the agent). This has a number of consequences. First, it relates back to the *curse of dimensionality* mentioned earlier, the fact that an agent’s state space grows exponentially with the number of state variables to consider. This means that as state spaces grow larger, so does the agent’s search time for finding an optimal policy. This scenario is especially problematic for large-scale and complex systems such as the one we are addressing. Using a hierarchical information state to constrain the agent’s search space by prior knowledge of the domain, which is often obvious to a system designer (such as that decisions need not be considered more than once per instruction), can thus help to mitigate the problem of growing search spaces and allow the agent to scale up to larger domains. A second consequence of the gain in learning speed is that learning in real-time becomes feasible for some scenarios as such demonstrated by Cuayáhuitl and Dethlefs [2011a,b]. The Hierarchical IS could also be used to constrain the agent’s behaviour so as to avoid ambiguous references to objects in cases where a learning agent fails to learn this behaviour. Finally, the notion of a hierarchical Information State is code-independent which facilitates the development of adaptive NLG systems using constraints.

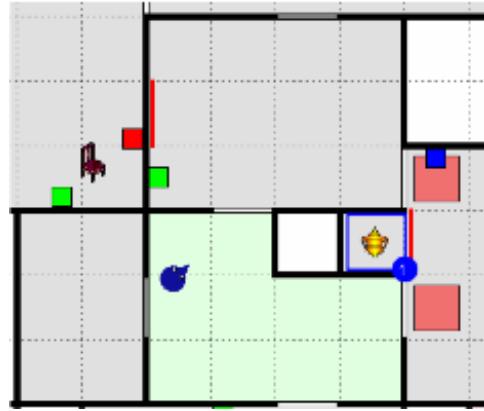
4.6.2 Human Rating Study

Even though the simulation-based results are encouraging, in this section we would like to investigate the effect that different instructions have on human ratings. In particular, we will compare three types of instructions: (1) human instructions that were drawn randomly from the GIVE corpus, (2) instructions that were generated using decision trees (we call them *deterministic* in this section), and (3) instructions that were generated using the reinforcement learning policy (which we will call *learnt* in this section). In this setting, we use the semi-learnt policy, but recall that both the fully-learnt and the semi-learnt policies eventually learn an equivalent behaviour.

4. Experimental Results

Please rate each instruction for its helpfulness on a scale of 1 to 5, where 5 represents the best score and 1 the worst.

- (a) Go the the room on the left. Press the green button.
- (b) Go to the green button. Press it.
- (c) Go to the next room and press green.



Please circle the intended referent.

Figure 4.5: Example of a spatial scene matched with one learnt (a), one deterministic (b) and one human (c) instruction. Participants were asked to rate each instruction for its appropriateness and circle the intended referent.

Figure 4.5 shows an example of the spatial scenes we used for the human rating study. Each spatial scene (in which the user is always depicted as a little blue head) is paired with one human, one deterministic and one learnt instruction. Instructions consist of a navigation instruction (to a referent button) followed by an instruction to press the button in the form of a referring expression. Participants in our study were asked to rate instructions on a 1-5 Likert scale (where 5 represents the best value and 1 the worst) for their appropriateness and helpfulness of guiding the displayed user from their origin to pressing the intended referent. Finally, participants were asked to circle the button which they considered the intended referent. The evaluation contained five different types of spatial scenes:

1. scenes that contain only one button,
2. scenes that contain two buttons, the referent and a distractors of the same colour as the referent,
3. scenes that contain two buttons, the referent and a distractors of a different colour than the referent,
4. scenes that contain two buttons, the referent and a same-coloured distractor, and an additional landmark (as shown in Figure 4.5), and

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

5. scenes that contain two buttons, the referent and a different-coloured distractor, and an additional landmark.

All scenarios occurred twice in each evaluation sheet, their specific instances were chosen from the GIVE corpus randomly. Scenes and instructions were presented to the users in a randomised order.

On the whole, 11 participants, 6 female and 5 male with an age average of 26.4, took part in the rating study, and rated altogether 132 sets of instructions. The *human* instructions were rated with an average of 3.82 with a standard deviation (*SD*) of 0.93. The *learnt* instructions were rated with an average of 3.55 (*SD* = 1.08) and the *deterministic* instructions (based on decision trees) were rated with an average of 2.39 (*SD* = 0.83).¹ According to a paired two-tailed t-test, the difference between *human* and *learnt* is not significant ($p < 0.40$) and has a small effect size $r = 0.13$. In contrast, the difference between *human* and *deterministic* is significant at $p < 0.002$ with a moderate effect size $r = 0.62$, and the difference between *learnt* and *deterministic* is significant at $p < 0.003$ with $r = 0.51$. Participants were able to identify the intended button referent in 96% of all cases.

4.7 Conclusion

In this chapter, we have developed the notion of a hierarchical information state and integrated it into our hierarchical reinforcement learning framework. The hierarchical IS was used as a principled mechanism to constrain the agent's action selection process in order to correspond to human preferences (induced using supervised learning) or prior knowledge of the domain that is obvious to the system designer. An evaluation showed two main effects of the constrained learning framework. First, we have seen that using constraints, we can still learn an optimal policy that is equivalent to a fully-learnt policy that was not subject to constraints during learning. Second, we have seen that using prior knowledge to constrain the agent's search space accelerates learning substantially so that

¹The main reason for the bad performance of the deterministic baseline was that it displayed a strong preference for destination instructions as learnt by the decision trees. This was not felicitous in all circumstances.

4. Conclusion

the framework scales to larger search spaces and we are able to address more complex domains of application. While the effect of faster learning can also be observed for a flat learning setting, i.e. flat reinforcement learning with a single information state [Heeman, 2007], it is stronger for the divide-and-conquer approach suggested in this thesis. The reason is that reinforcement learning with a hierarchical setting is per se more scalable than using a flat setting.

In order to test the effect that the instructions generated using our semi-learnt policy have on human ratings, we compared them against a human baseline and a supervised learning baseline that was obtained using decision trees trained from the GIVE corpus data. Human judges were shown a graphical spatial scene together with one human instruction, one semi-learnt instruction (using hierarchical RL with a hierarchical IS) and one instruction generated using decision trees. The participants were asked to rate each instruction for its appropriateness and helpfulness. Results showed that the semi-learnt instructions generated using our framework were rated only slightly worse than the human instructions. The difference was statistically not significant. In contrast, instructions generated using decision trees were rated significantly worse than both human and semi-learnt instructions.

While this chapter has addressed decisions of content selection in some detail, an open question so far remains how we can improve the optimisation of surface realisation further. So far the agent merely produces grammatical forms that correspond to the users' need of detail and information during generation. However, we would like it to produce surface forms that have a high likelihood of occurrence according to a language model of the domain, while still preserving their other advantages. In the next chapter, we will address this topic and present an approach that integrates graphical models—to represent the agent's generation space—into our constrained joint learning framework.

4. A HIERARCHICAL INFORMATION STATE FOR CONSTRAINED LEARNING

Chapter 5

Graphical Models for Surface Realisation

5.1 Introduction

Surface realisation components of NLG systems typically face a non-determinacy between a semantic concept and its (many) possible syntactic and lexical realisations. Even when given a human corpus of the target domain and a semantic content to communicate, there is usually more than one way to express the desired semantics. One way to address this problem in recent years has been to formalise the set of surface realisation options into a *generation space*, that is a systematic space of realisations for a semantic concept. A generation space can take different forms that come with different styles and applications. [Belz \[2008\]](#) suggests to use probabilistic context-free grammars (PCFGs) as generation spaces for weather forecasts with variable surface forms. [Georgila et al. \[2002\]](#) use Hidden Markov Models (HMMs) for recognition and generation in spoken dialogue systems and [Dethlefs and Cuayáhuitl \[2011a\]](#) use them to generate navigation instructions in a hierarchical reinforcement learning framework. [Mairesse et al. \[2010\]](#) use Bayesian Networks (BNs) for surface realisation within an Active Learning framework and [Dethlefs and Cuayáhuitl \[2011b\]](#) use them to generate navigation instructions and referring expressions in a joint learning framework. [Barzilay and Lee \[2002\]](#) use multiple sequence alignment to obtain lattices of surface form variants for a semantic concept in the domain of mathematical proofs. All of these approaches have shown that generation spaces can mostly be induced automatically from data and can achieve good surface realisation results. A main advantage arising from the use of generation spaces over deterministic surface realisation is the variable system output that can make generated language seem more lively and natural than deterministic output. Studies involving human participants have shown that humans prefer variation in system output [[Belz and](#)

5. GRAPHICAL MODELS FOR SURFACE REALISATION

Reiter, 2006; Foster and Oberlander, 2006] even if it is just for variation’s sake. Variation should not be random, though. In Chapter 3 we have seen that while our learning agent was able to generate grammatical and variable surface forms, the randomly varied surface forms were awkward in a number of cases and difficult to comprehend. A better motivated and controlled form of variation is therefore needed. In this chapter we will focus on surface realisation and on providing our learning agent with a principled and well-motivated means of choosing among surface realisation variants. We will argue that surface realisation choices should be made with respect to three trade-offs: (1) they should have a high likelihood according to a language model of the domain, (2) they should display variation in an appropriate proportion, and (3) they should be jointly optimised with choices of content selection and utterance planning.

In the remainder of this chapter we will first analyse human corpus data and look for evidence of the proportion with which humans use variation in their language. We will then investigate a range of different graphical models for the representation of generation spaces and compare them based on their theoretical and practical properties. In this context, we will focus on context-free and probabilistic context-free grammars, HMMs and Bayesian Networks. We will integrate the graphical models into our joint learning framework and compare their performance in conjunction with content selection and utterance planning. An evaluation will compare the different graphical models based on their average rewards in simulation and the similarity of surface forms with human authors.

5.2 Variation and Alignment in Human Data

In psycholinguistics and psychology, *alignment* describes the process by which interlocutors adapt to each other’s linguistic representations through gradual entrenchment. Pickering and Garrod [2004] define the concept of *interactive alignment*, that is alignment during the course of an interaction, in the following citation:

‘The interactive alignment model assumes that successful dialogue involves the development of aligned representations by the interlocutors. This occurs by

5. Variation and Alignment in Human Data

priming mechanisms at each level of linguistic representation, by percolation between the levels so that alignment at one level enhances alignment at other levels, and by repair mechanisms when alignment goes awry. ’ [Pickering and Garrod, 2004], p. 7

Under this perspective, interlocutors align their situational and linguistic representations during dialogue through an automatic priming mechanism and engage in a negotiative repair process whenever their representations appear incompatible. Apart from interactive alignment, speakers often self-align with themselves. A possible explanation is that humans use the same mental representations during language production and comprehension, so that alignment occurs regardless of whether the last utterance was made by another person or by the speaker him- or herself [Garrod and Anderson, 1987; Pickering and Garrod, 2004]. The dialogue displayed in Figure 5.1, where two people describe their position in a maze, shows examples of both types of alignment (highlighted in blue). In the first example, ‘extreme right’, speaker B aligns with the phrase that speaker A had previously suggested. In the second example, ‘right indicator’, speaker B self-aligns in two consecutive utterances. Both examples, or *dialogue routines* as they are also called, are cases of lexical alignment. Importantly however, the interactants here also align their description schemes of the maze. While we can straightforwardly locate A’s position in the maze at the ‘right indicator’, it is by no means obvious that this expression is the only, or even best, way to describe the situation at hand. This conceptualisation is therefore local to the current dialogue and might not persist beyond its duration. Pickering and Garrod [2005] interpret dialogue routines with respect to Jackendoff’s notion of a lexicon [Jackendoff, 2002], which can contain anything from a single word over complex phrases and idioms up to arbitrarily complex memorised pieces of language.

Alignment does not just occur at the lexical level, though. It has been observed at all linguistic levels in a variety of experimental studies. Lexical alignment in the manner demonstrated for the maze dialogues above was shown by Garrod and Anderson [1987], Garrod and Clark [1993] and Garrod and Doherty [1994]. Similarly, Brennan and Clark [1996], Clark and Wilkes-Gibbs [1986] and Wilkes-Gibbs and Clark [1992] presented studies in which interactants continu-

5. GRAPHICAL MODELS FOR SURFACE REALISATION

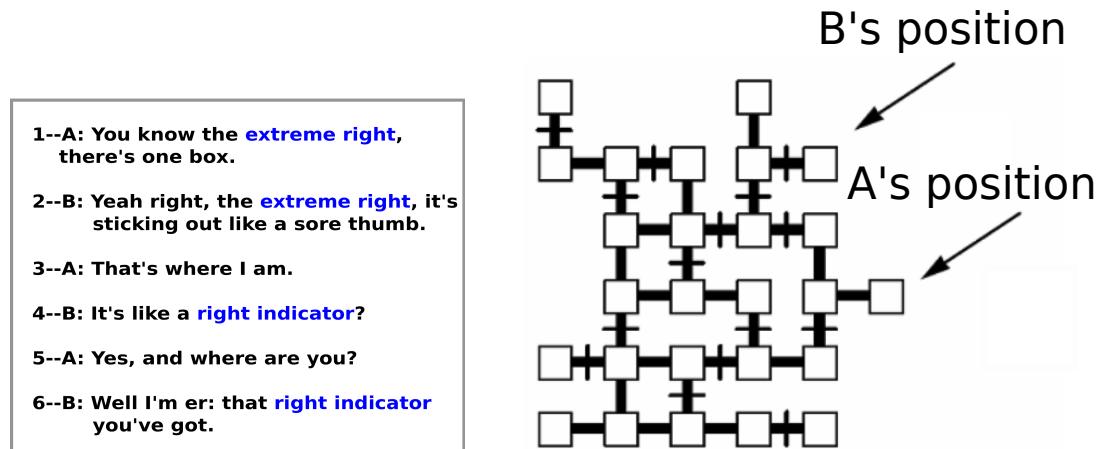


Figure 5.1: Extract from a dialogue of the maze protocols exemplifying two cases of alignment (shown in blue). In the first case, ‘extreme right’, speaker B aligns lexically with speaker A. In the second example, ‘right indicator’, speaker B self-aligns. The example was adapted from Pickering and Garrod [2006], p. 5–6.

ously adjusted their referring expressions for objects in various matching tasks. Syntactic alignment was demonstrated by Bock [1986] and later by Branigan et al. [2000]. The latter study used a picture naming task involving verbs that could either occur in a double object construction or in a construction containing a prepositional phrase, e.g. *hand* in ‘*hand the pirate a cake*’ or ‘*hand a cake to the pirate*’. It was shown that a confederate using either one construction would raise the likelihood of the participant using the same construction subsequently. Haywood et al. [2005] were able to show that this priming effect was strong enough to even override a tendency to disambiguate phrases such as ‘*Put the penguin on the star* [...]’ in contexts where ‘*on the star*’ could either refer to the penguin’s current location (which could be disambiguated as ‘*the penguin that is on the star*’) or its desired location subsequent to putting it there. An interesting study for system developers in this context may also be Levelt and Kelter [1982] who show that humans in fact prefer conversations with other humans who align with them over those that do not align. This may also be true for conversations with interactive systems.

5. Variation and Alignment in Human Data

5.2.1 Variation and Alignment in the GIVE Corpus

With the background of interactive alignment established, let us have another look at the GIVE corpus dialogues. There are numerous cases of alignment here. First of all, we have already observed cases of semantic alignment in the previous chapter, where instruction givers used a predominantly consistent navigation strategy throughout. We can also find cases of lexical and syntactic alignment, though, some of which are shown in Table 5.1. The aligned phrases here are shown in bold-face and the number of instructions intervening between aligned instructions are given in parentheses. In the first example, the instruction giver uses the phrase ‘*you want*’ with high frequency and across instruction types. This is especially interesting since the phrase per se has a rather low frequency in the corpus on the whole (1.8% of all verbs). In the second example, the instruction giver has a preference for using path instructions including the verb ‘*go*’. We can observe numerous uses of the phrases ‘*go out*’ and ‘*go through*’, which again are used with a substantially higher frequency in this discourse (31.6%) than in the GIVE corpus overall (12.2%). In the third example, the instruction giver produces referring expressions almost exclusively using the verbs ‘*click*’ (33.3% in this dialogue and 33% in the entire corpus) and ‘*hit*’ (66.6% in this dialogue, 6.6% in the corpus). Other verbs that have a high likelihood in the GIVE corpus and with other instruction givers, such as ‘*push*’ (21.6%) or ‘*press*’ (also 21.6%), are entirely absent from this dialogue.

Apart from the observation that instruction givers self-align with their own surface forms, though, we can also see that alignment is not absolute. Human instruction givers both align with and vary their own surface forms. A possible explanation could be that they prefer variation over total alignment for stylistic reasons in accordance with studies that show that humans prefer variation for variation’s sake [Belz and Reiter, 2006; Foster and Oberlander, 2006]. An alternative explanation is that variation is random and just as automatic and subconscious as alignment. Levelt [1989] formulates the following hypothesis concerning human language production:

‘Grammatical encoding takes a message as input and delivers a surface struc-

5. GRAPHICAL MODELS FOR SURFACE REALISATION

Examples of Alignment in the GIVE Corpus

(1) Lexical Alignment Across Instruction Types

... (15 instructions)

great, **you want** to press that green button ... (1 instruction)

you want to press the yellow on the wall to the left first ... (3 instruc.)

you wanna get that red button ... (1 instruction)

now **you want** to get the blue button ... (9 instructions)

you want to exit the room you are in ... (17 instructions)

you want to keep going straight but to the left ... (25 instructions)

okay **you want** to take a left ... (13 instructions)

(2) Lexical Alignment in Path Instructions

... (1 instruction)

see the opening on your right and **go straight through** it ... (2 instruc.)

ok **go through** the one you are facing ... (2 instructions)

go through the door that just opened ... (3 instructions)

go through to the left ... (2 instructions)

go out and press the blue button to the right

go back into the previous room and press the button again

go out to your right ... (2 instructions)

go through the opening

go through the door behind you ... (1 instruction)

go out and press the green button on the right

go through the opening ... (1 instruction)

go out

go out right (2×)

(3) Lexical alignment in Referring Expressions

ok, **hit** the blue button on the wall behind you ... (11 instructions)

click the green button ... (1 instruction)

click the yellow button on the wall

now **click** the red button directly behind you ... (1 instruction)

now **hit** the blue one directly next to the yellow one

ok, **hit** the other red button, closer to the opening ... (4 instructions)

hit the green button near the couch ... (1 instruction)

and **hit** the red button ... (4 instructions)

hit the yellow button ... (7 instructions)

Table 5.1: Three examples of (self-)alignment in the GIVE corpus. In the first example, the instruction giver uses the phrase ‘*you want*’ with high frequency and across instructions. In the second example, the instruction giver has a preference for using path instructions including the verb ‘*go*’. In the third example, the instruction giver uses exclusively the verbs ‘*click*’ and ‘*hit*’ in their referring expressions. The number of intervening instructions are shown in parentheses behind each instruction.

5. Variation and Alignment in Human Data

ture as output. It is likely that this process is highly automatic and nonintentional. A speaker will not, for every message, consider which of various grammatical alternatives would be most effective in reaching some communicative goal. ’ (Levelt [1989], Chapter 7, p. 282)

We will not investigate the question here of why variation (or alignment) occurs in human discourse. Rather we will take the stance that if it occurs as ubiquitously as we have observed in our human data, then our hierarchical learning agent should be able to model it.

5.2.2 A Constituent Alignment Score

In order to determine an appropriate proportion of alignment and variation for our learning agent, we follow Dethlefs and Cuayáhuitl [2010] who compute the proportion of alignment and variation of human-authored route instruction texts. They perform an analysis of 24 human route instructions based on the degree of lexical repetition in the texts. Their analysis is based on Hirst and St-Onge [1998] who retrieve lexical chains from texts by identifying a number of relations between lexical items. The analysis was based on Hirst and St-Onge’s *extra-strong relations* (which are based on the repetition of a lexical item) since these can be computed from shallow properties of texts and do not require expensive annotations. Assuming that each lexical phrase (or constituent) that has been used before is part of a lexical chain, we can compute the *constituent alignment score (CAS)* for every constituent in the discourse. This is a score that indicates the proportion of alignment and variation (in the discourse so far). It is computed as

$$CAS = \frac{\text{Lexical Tokens in Chains}}{\text{Total Number of Tokens}} \quad (5.1)$$

and yields a number in the range of $[0 \dots 1]$. Using this equation, Dethlefs and Cuayáhuitl [2010] obtain an average alignment score of 0.43 for the 24 texts. According to this score, humans seem to vary and align their surface forms in an about equal proportion—at least in the navigation domain. For our learning agent we will therefore target a *CAS* of roughly 0.5. This target score will be

5. GRAPHICAL MODELS FOR SURFACE REALISATION

included in the reward function in Section 5.4.1. First of all, though, we will discuss and compare several ways of representing a generation space for surface realisation that can be integrated into our learning framework.

5.3 Representing Generation Spaces as Graphical Models

In this section we will compare three different models with respect to their suitability of representing generation spaces for surface realisation: probabilistic context-free grammars, Hidden Markov Models and Bayesian Networks.

5.3.1 (Probabilistic) Context-Free Grammars

Context-free grammars (or Phrase-Structure Grammars) are designed to describe small portions of language and can be used for analysis or generation. A central property of CFGs is their hierarchical structure which requires larger phrases to consist of smaller phrases. This principle is based on the notion of constituency. While some linguists follow a strict interpretation of constituency [Chomsky, 1957], in which constituents must be chosen in a way that yields an exclusively right-branching parse tree, others follow a more liberal interpretation in which nearly any unit can function as a constituent [Steedman, 2000]. We will follow the latter interpretation here since it allows us to learn CFGs from human corpus data automatically based on the notion of interchangability. Even though CFGs have originally been designed for parsing, we can also use them in the opposite direction, i.e. for generating sentences. In NLG, the idea of a generation space (that is represented as a CFG) is to group a set of sentences together that express the same semantic content. The generation space can then be consulted whenever a sentence is needed. A CFG can be defined as consisting of

- a start symbol N ,
- a set of terminal symbols $W = \{w_0, w_1, w_2 \dots w_{|W|}\}$, which correspond to words or phrases, and need to be specified as the *lexicon* of the CFG,

5. Representing Generation Spaces as Graphical Models

- a set of nonterminal symbols $N = \{n_0, n_1, n_2 \dots n_{|N|}\}$, which correspond to semantic categories and typically consist of smaller constituents or terminal symbols, and
- a set of production rules $H = \{h_0, h_1, h_2 \dots h_{|H|}\}$ of the form $n \rightarrow \alpha$, where $n \in N$, $\alpha \in W \cup N$. These rules are expanded in the process of generating sentences with the CFG.

Alternatively, we can represent a generation space as a Probabilistic context-free grammar (PCFG), the CFG’s probabilistic counterpart. An important advantage of PCFGs in parsing has been their capability to address ambiguity in language by using probabilities to infer the intended meaning. In generation, we can use probabilities to guide the choice of competing surface realisation variants. A PCFG consists of the same components as a CFG, except that its rules are of the form $n \rightarrow \alpha[\gamma]$, where γ is a number in the range $[0 \dots 1]$ expressing $Pr(\alpha|n)$. This is the conditional probability of an expansion α given the nonterminal symbol n .

To obtain a CFG (or a PCFG) from human data, we use the ABL algorithm [van Zaanen, 2000], which aligns a set of unlabelled input strings based on Minimum Edit Distance [Wagner and Fischer, 1974] and induces a CFG from the examples. We used the GIVE corpus (cf. Section 3.4.2) as an input to the algorithm to induce one CFG per annotated instruction type, i.e. *destination*, *direction*, *orientation*, *path*, ‘*straight*’ and *referring expression*. Note that these instruction types correspond to the surface realisation agents $M_{0\dots 5}^3$ (cf. Figure 3.9). For the PCFGs, a probability for each resulting grammar rule was estimated using Maximum Likelihood Estimation (MLE).

As an example of a resulting PCFG consider the grammar in Figure 5.2. We can see two types of rules here. One is of the form $n \rightarrow \alpha \beta [\gamma]$ and the other is of the form $n \rightarrow \alpha \mid \beta [\gamma]$. The first type of rule can be interpreted as a conjunction, i.e. the semantic concept n can be expanded into the terminal symbol α followed by the terminal symbol β . The second type of rule can be interpreted as a disjunction, i.e. here the semantic concept n can be expanded into either the terminal symbol α or into the terminal symbol β . Each rule has a probability of expansion attached to it which is shown in bold-face. In addition, each rule

5. GRAPHICAL MODELS FOR SURFACE REALISATION

Example PCFG for destination instructions

```
destination1 → desVerb1 desPrep1 desRel1 (detail) .5
destination2 → desVerb1 desPrep2 desRel2 .4
destination3 → desVerb2 desRel1 | desRel2 .1
desVerb1 → go .89 | keep-going .02 | walk .05 | return .01 |
            continue .001 | head .03 | move .001 | empty .001 1.0
desVerb2 → you_need .05 | you_want .99 | get .05 1.0
desPrep1 → to .56 | towards .32 | until .12 1.0
desPrep2 → into .98 | in .02 1.0
desRel1 → pointRelatum 1.0
desRel2 → roomRelatum 1.0
detail → direction .1 | straight .1 | empty .8 1.0
```

Figure 5.2: Example PCFG for destination instructions. Nonterminal symbols represent semantic constituents and terminal symbols represent possible realisations. Probabilities of occurrence for individual surface forms are given behind the surface form, the probability of expansion for each rule is given in bold-face.

containing a disjunction has a probability for each terminal symbol that it can be expanded into. Note that we can use the representation shown in Figure 5.2 as both a CFG and a PCFG. For the former case, we can simply ignore the probabilities and apply production rules randomly. The example in Figure 5.2 shows a possible PCFG for generating destination instructions. They can be phrased in three different ways. Type 1 ('destination1') generates instructions such as 'Go to the sofa' (referring to point-like destinations) and Type 2 ('destination2') generates instructions such as 'Go into the next room' (referring to room-like destinations). Type 3 ('destination3') uses verbs followed directly by either type of relatum, e.g. you need the door'. Instructions can be augmented with optimal detail (last rule 'detail'), such as the direction of the destination. See Appendix A for more details on the realisation options of destination instructions. For more information on (probabilistic) context-free grammars, please refer to [Jurafsky and Martin \[2009\]](#), Chapters 12, 13 and 14.

5.3.2 Hidden Markov Models

Hidden Markov Models can be seen as sequence classifiers that assign a label to each unit in a particular sequence. Assume for example that we were interested

5. Representing Generation Spaces as Graphical Models

in the weather on a sequence of days in the distant past before thermometers were invented. The only data available to infer the weather on these days is the number of ice creams eaten by a specific individual.¹ In addition, we know the probability of a hot day being followed by another hot or a cold day (and vice versa). We can define an HMM formally as consisting of the following components (see [Jurafsky and Martin \[2009\]](#), Chapter 6).

- a set of *states* $S = \{s_0, s_1, s_2 \dots s_{|S|}\}$ that capture the knowledge of the HMM, or the possible classification values, and where we denote a state sequence as $i_0 \dots i_n$ with i_t being the state at time t ;
- a set of *observations* $O = \{o_0, o_1, o_2 \dots o_{|O|}\}$ that are drawn from a finite vocabulary $V = \{v_0, v_1, v_2 \dots v_{|V|}\}$ of elements that the HMM can observe as input from the environment, and where the observation o at time t is denoted as o_t ;
- a *transition probability matrix* $A = a_{01}, a_{02} \dots a_{n1} \dots a_{nm}$, where each a_{ij} represents the probability of moving from one state i to another state j , s.t. $\sum_{j=1}^n a_{ij} = 1 \forall_i$, and
- a sequence of *observation likelihoods* $B = b_i(o_t)$, also called *emission probabilities*, where each one expresses the likelihood of an observation o_t being generated from a state i .

Optionally, we could add a start and an end state. With these components defined, let us return to our weather example. We can formulate states that correspond to the weather events ('hot' and 'cold') and observations that correspond to the number of ice creams eaten. The probability of a hot day being followed by another hot or a cold day (or vice versa) and the probabilities of observing a hot or a cold day are defined as the transition and emission probabilities, respectively. The resulting HMM is shown on the left of Figure 5.3. Trained on a sufficient amount of data, the HMM could now learn to make informed guesses about weather events in the designated distant past of the example.

A popular application of HMMs to Natural Language Processing (NLP) has been in the area of Part-of-Speech (POS) tagging. The task here is to observe a

¹This example is adapted from [Jurafsky and Martin \[2009\]](#).

5. GRAPHICAL MODELS FOR SURFACE REALISATION

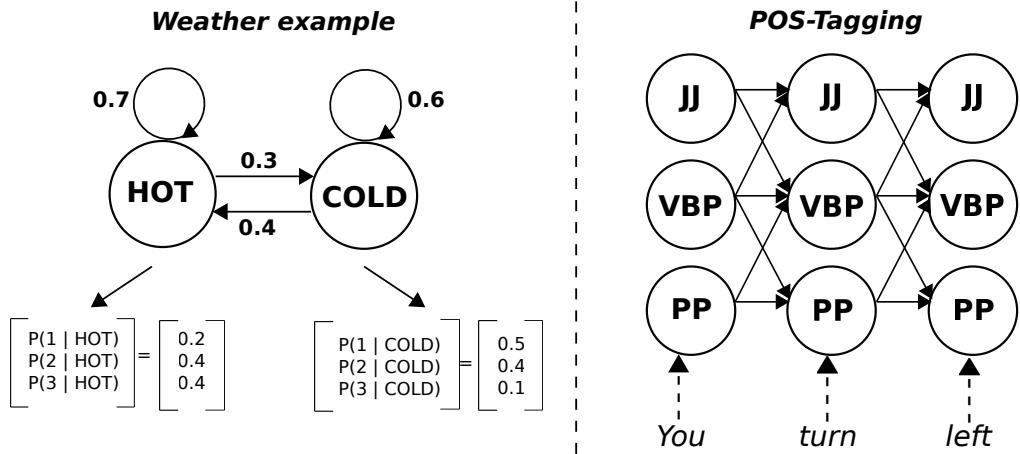


Figure 5.3: Example Hidden Markov Models for predicting weather events (Jurafsky and Martin [2009], Chapter 6) on the left-hand side, including transition and emission probabilities, and for Part-Of-Speech Tagging for the sentence ‘*You turn left*’.

sentence in natural language and infer its most likely corresponding sequence of POS tags. The context in which a word occurs in a sentence can give important information on its most likely POS tag. Possessive pronouns in English, for example, are most often followed by a noun phrase. Personal pronouns, on the other hand, are followed by a verb phrase (Jurafsky and Martin [2009], Chapter 5). As an example illustrating the application of HMMs to POS-tagging, see the right-hand side of Figure 5.3. The states here correspond to POS tags and the observations to words. Transition and emission probabilities are typically learnt from annotated training data. In the figure, we see a left-to-right HMM for POS tagging the sentence ‘*You turn left*’ based on the Penn Treebank Tagset [Marcus et al., 1993].

The idea of representing a surface realiser as a left-to-right HMM can be roughly defined as the converse of POS tagging. While in POS tagging the task is to map an observation string of words onto a hidden sequence of POS tags, in surface realisation we face the opposite scenario. Given an observation sequence of semantic symbols, we want to map it onto a hidden most likely sequence of words. To use HMMs as representations of generation spaces, we can treat states as representing words or phrases so that a sequence of states

5. Representing Generation Spaces as Graphical Models

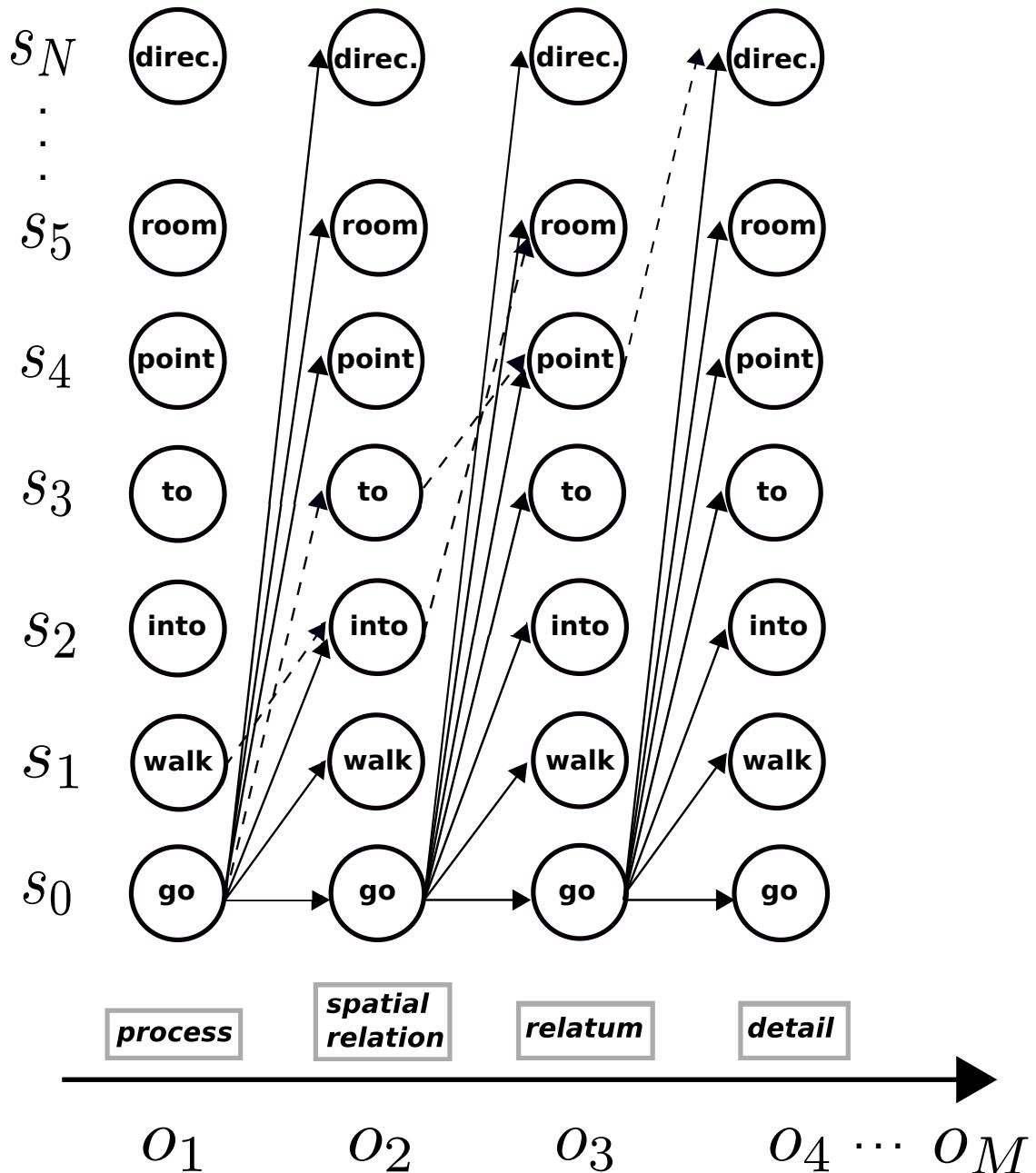


Figure 5.4: Example trellis for an HMM representing the generation space of destination instructions. The observation sequence corresponds to those semantic concepts that need to be realised. States correspond to possible surface realisations of the semantic concept. Note that only a subset of all states and transitions are shown. Dashed arrows correspond to sequences that occur in the data.

5. GRAPHICAL MODELS FOR SURFACE REALISATION

$i_0 \dots i_n$ represents a sentence (or a longer phrase). An observation sequence $o_0 \dots o_n$ can be defined as representing a finite set of semantic concepts. Each symbol has an observation likelihood $b_i(o)_t$, which gives the probability of observing o in state i at time t . We again use one HMM per instruction type of *destination*, *direction*, *orientation*, *path*, ‘*straight*’ and *referring expression*. Figure 5.4 shows an example trellis for an HMM for a destination instruction. For reasons of space not all states and transitions are shown. The HMM was constructed from the PCFG shown in Figure 5.2. Each nonterminal symbol of the automatically induced CFG corresponds to a possible observation of the HMM and each terminal symbol of the CFG corresponds to one HMM state. The transition and emission probabilities of the HMM were learnt from the GIVE corpus annotations during training using the Baum-Welch algorithm. An implementation of this algorithm is available as part of the JAHMM toolkit.¹ For more details on HMMs, please see Bishop [2006]; Rabiner [1989]; Rabiner and Juang [1993]; for more details on HMMs in NLP see Jurafsky and Martin [2009], Chapters 5 and 6.

5.3.3 Bayesian Networks

Bayesian networks model the dynamics between a set of random variables and their values. Consider first the example discussed by Charniak [1991] and illustrated by the Bayesian Network in Figure 5.5. We are here trying to predict whether the family is home before trying the door. To make the prediction, several pieces of evidence are available. For example, when the family goes out, they often switch on the outdoor light and put out the dog. However, they also switch on the light when expecting a visitor, and they put out the dog when she has a bowel problem. Further, when the dog is out, we should hear her bark but we could also be confused by the barking of the neighbour’s dog. Bayesian Networks can be applied to problems where a causality seems to play a role but we have only incomplete knowledge of the domain. For example, what should we conclude when the dog is out, but the light is not on? Or when the light is on, but the dog is inside? Because knowledge of all influencing factors (and their combinations) is typically incomplete, especially for large problems, Bayesian Networks allow us

¹<http://code.google.com/p/jahmm/>

5. Representing Generation Spaces as Graphical Models

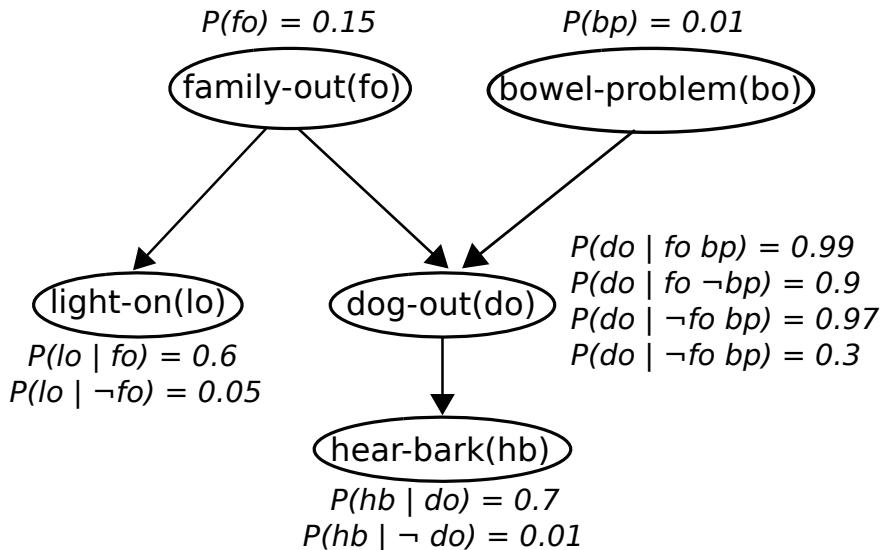


Figure 5.5: Example Bayesian Network for predicting whether the family is home or not, adapted from Charniak [1991], including the conditional probability tables over random variables. The nodes (random variables) denote states of affairs and the arcs (uni-directional) causal connections.

to perform probabilistic inference on a problem and make informed predictions. They are directed acyclic graphs with random variables that are partially independent of each other. Random variables are associated with a prior probability distribution or learnt parameters (probabilities), so that conditional probabilities can be computed for combinations of random variables (discrete in our case). A Bayesian Network can be defined by

- a set of random variables $Y = \{Y_0, Y_1, Y_2 \dots Y_{|Y|}\}$, each of which can take a value out of a set of values with associated probabilities; they express our domain knowledge;
- a set of parents $pa = \{pa(Y_0), pa(Y_1), pa(Y_2) \dots pa(Y_{|Y|})\}$ of random variables, where each random variable Y_j depends only on its parents $pa(Y_j)$,
- a set of conditional probability distributions of the form $P(Y_j | pa(Y_j))$, where each random variable Y_j is associated with a unique conditional probability distribution depending on itself and its parents $pa(Y_j)$. For variables that have no parents, the prior probability of the variable is used instead.

5. GRAPHICAL MODELS FOR SURFACE REALISATION

We can compute a unique joint probability distribution $P(Y)$ for the set of random variables Y according to

$$P(Y) = \prod P(Y_j | pa(Y_j)), \quad (5.2)$$

where $pa(Y_j)$ denotes the set of parents of Y_j , and every variable is associated with a conditional probability distribution $P(Y_j | pa(Y_j))$. A key property of Bayesian Networks is the assumption that variables are *conditionally independent* of their non-descendents given their immediate parents. We can say that variable A is conditionally independent of variable B given C if $P(A, B | C) = P(A | C)P(B | C)$ such that $P(C) \neq 0$ [Ghahramani, 2001]. This property is called the *d-separation* criterion. Given a Bayesian Network defined in this way and its probabilities such as shown in Figure 5.5, we are in a position to do inference in order to answer probabilistic queries.

A generation space defined as a Bayesian Network gives us the opportunity to model the dynamics between a semantic concept and its possible surface realisations. Based on the CFGs we induced from human corpus data, we consider random variables to correspond to semantic concepts (i.e. the nonterminal symbols in the CFG or the observation symbols in the HMM). Correspondingly, we design the values of random variables to represent words or phrases, the surface realisation variants, of the semantic concept (i.e., the terminal symbols of the CFG or the states of the HMM). We design one Bayesian Network per instruction type for *destination*, *direction*, *orientation*, *path*, ‘*straight*’ and *referring expression*. Figure 5.6 shows an example Bayesian Network for destination instructions. It contains two main types of dependencies. First, the random variable ‘information need’ influences the inclusion of optional semantic constituents (‘destination detail’) and the process of the utterance (‘destination verb’). Second, a sequence of dependencies spans from the verb to the end of the utterance. In Figure 5.6, this is from the verb over the preposition to the relatum. The first dependency is based on the intuition that whenever the user’s information need is high, optional semantic information is more likely to be included than when the information need is low.¹ We assume that high frequency verb forms are preferable in cases of a

¹This is key to the joint treatment of content selection and surface realisation: if an utterance is not informative in terms of content, it will receive bad rewards, even with good surface

5. Representing Generation Spaces as Graphical Models

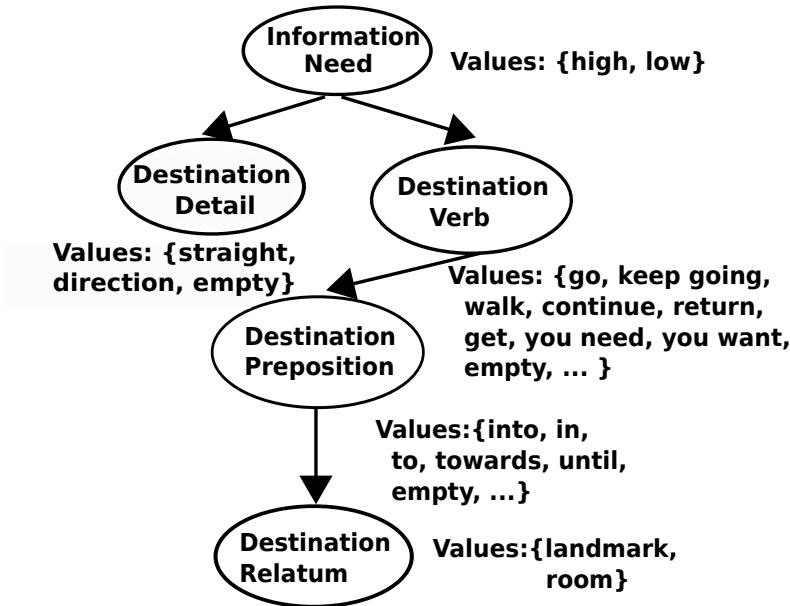


Figure 5.6: Example Bayesian Network representing the generation space of destination instructions. Random variables correspond to semantic concepts and their values correspond to possible surface realisations of the semantic concept.

high information need. The second dependency is based on the hypothesis that the value of one constituent can be estimated based on the previous constituent. Since Bayesian Networks allow for probabilistic reasoning, that is the calculation of posterior probabilities given a set of observed variable-value pairs, we can perform reasoning over surface forms. Given the word sequence represented by lexical and syntactic variables $Y_0 \dots Y_n$, and situation-based variables $Y_{n+1} \dots Y_m$, we can compute the posterior probability of random variables Y . We use an efficient implementation of the variable elimination algorithm [Cozman, 2000] for probabilistic reasoning that is available as part of the JavaBayes toolkit.¹ The parameters of the Bayesian Networks were estimated using MLE with Witten-Bell smoothing. Please see Bishop [2006]; Charniak [1991]; Jensen [1996] for a more detailed discussion of the properties of Bayesian Networks.

realisation choices (and vice versa).

¹<http://www.cs.cmu.edu/~javabayes/>

5. GRAPHICAL MODELS FOR SURFACE REALISATION

5.3.4 Comparison of Graphical Models

Generation spaces using different formalisations come with different properties. We have seen that CFGs were easy to obtain and manage. They can be automatically induced from unlabelled data and are fully functional representations of generation spaces for random variation. Unfortunately, their choices are not sensitive to either the properties of the environment (such as the spatial situation, the user, or decisions of content selection and utterance planning) nor to the likelihood of different surface form variants according to the language model. CFGs will represent the *random choice* baseline for our later evaluation in Section 5.5.1.1.

PCFGs share all properties of CFGs but are in addition sensitive to the likelihood with which different surface form variants occur in the human data. This property can be used in several ways. We can, for example, aim to generate the most likely sentence overall. Alternatively, we may want to always expand the most likely grammar rule (which does not necessarily lead to the most likely sequence overall); or we may wish to choose rules according to their probability so that a rule with a likelihood of 0.8 is chosen 80% of the time. Even though PCFGs allow a certain degree of controlled variation, they are still not able to correspond to properties of the environment or the user. PCFGs will represent our *greedy choice* baseline for the later evaluation.

HMMs and Bayesian Networks share a number of properties so that a comparison of their performance is less obvious. HMMs are able to observe properties of the environment and make decisions based on their observations. Furthermore, they have a notion of the likelihood with which different surface realisations occur in the human data and they can use them to motivate their own surface realisation decisions. Given these properties, the HMMs represent a more adaptive model of a generation space than CFGs or PCFGs. It therefore seems likely that they may also achieve a better performance in the later evaluation than the previous two models. HMMs are often faced with limitations due to their conditional independence assumptions: (a) the Markov assumption, (b) the stationary assumption, and (c) the observation independence assumption. Even though this property usually makes HMMs easier to train, maintain and scale than more

5. Representing Generation Spaces as Graphical Models

complex models such as Bayesian Networks (which need to take a number of dependencies across random variables into account for computation), it can also put them in a disadvantage with respect to context-awareness and accuracy as is e.g. observed by Olivier and Horvitz [2005]. See Rabiner [1989], Rabiner and Juang [1993], pp. 365, and Huang et al. [2001], p. 409, for discussions of the limitations of HMMs and Ghahramani [2001] for a discussion of different approaches to address these limitations, such as Factorial HMMs, Tree-structured HMMs, switching state-space models or approximate inference techniques.

The most context-aware generation space model are the Bayesian Networks. Their random variables allow them to keep a detailed model of the environment, the user, and the content selection and utterance planning choices relevant to the interaction. They are therefore able to compute the likelihood of a surface form not just based on their likelihood of occurrence in the corpus data, such as the other models, but with respect to all relevant properties of the current situation. While Bayesian Networks also place conditional independence assumptions on their variables, they can overcome the problem of lacking context-awareness that HMMs face by the dependencies they add across random variables. A further advantage of Bayesian Networks is that their *d-separation* criterion allows the conditional independence properties of the joint probability distribution to be read directly from the graph requiring no further analysis [Bishop, 2006]. Based on these advantages, we may predict that Bayesian Networks will contribute to the most sophisticated system behaviour in the evaluation in Section 5.5.1.1. However, Bayesian Networks also face a number of limitations. Given the dependencies they postulate among random variables, they can be more data intensive and less scalable than models with simpler structures (such as HMMs). This can be particularly problematic for large domains such as many real world applications. Several approaches have been suggested as a remedy for this problem including Object-Oriented Bayesian Networks [Koller and Pfeffer, 1997], Relational Bayesian Networks [Jaeger, 1997], Hierarchical Bayesian Networks [Gyftodimos and Flach, 2002], Markov Logic Networks [Richardson and Domingos, 2006] and Dynamic Bayesian Networks [Ghahramani, 1997; Murphy, 2002], which Bayesian Networks and HMMs converge into.

Since the structure of the Bayesian networks and HMMs in this chapter was

5. GRAPHICAL MODELS FOR SURFACE REALISATION

manually specified, the question of the best structure for a graphical model for surface realization arises. It is likely that different structures can have different effects on the results and performance of each model. A further consideration that is typically relevant to any machine learning approach is the issue of data sparsity. We need to ensure that enough training data is available for the graphical models to learn a robust behaviour. While both issues are important for the application of graphical models, we will not consider them further in this thesis.

5.4 Experimental Setting

This section will describe the experimental setting for an empirical comparison of different graphical models for surface realisation within a hierarchical learning framework. The full state-action space for the surface realisation agents, with a *Constituent Alignment Score* integrated into each agent, is provided in Appendix C. The simulated environment will again be used as before.

5.4.1 Integrating Surface Realisation: A Three-Dimensional Reward Function for Situated NLG

The reward function designed in the previous two chapters focused on optimising decisions of content selection and utterance planning. The final reward function that will be used to evaluate the approach designed in this thesis has three dimensions: (1) one for rewarding content selection and utterance planning decisions, (2) one for rewarding surface realisation decisions and (3) one for optimising the proportion of alignment and variation in system utterances.

5.4.1.1 Dimension 1: User Satisfaction

The first dimension, for the optimisation of content selection and utterance planning decisions, will again be used as before. The function

$$\text{Performance} = 0.38\mathcal{N}(GTS) - 0.87\mathcal{N}(UT), \quad (5.3)$$

rewards *the shortest possible interaction at maximal task success*. It rewards

5. Experimental Setting

the actions of agents $M_0^0 \dots M_4^2$ by computing a numeric reward after each fully generated utterance and its following user reaction. This reward is back-propagated (by the learning algorithm) to all agents that contributed to the generated utterance. In addition, we assigned a reward of +1 one each navigation instruction whose navigation strategy (out of *low level*, *high level* and *mixed*) was consistent with the strategy of the previous instruction. This principle was found to correspond to human behaviour and was therefore considered worth modelling.

5.4.1.2 Dimension 2: Naturalness

To generate natural surface forms, the second dimension focuses on surface realisation and offers the opportunity to compare different graphical models according to the informativeness of the rewards that they assign as feedback to our learning agent. Agents $M_{0..5}^3$ will use a notion of likelihood to reward the learning agent for having generated word sequence $w_0 \dots w_n$ by assigning a reward $P(w_0 \dots w_n)$. This reward corresponds to different functions in different graphical models:

$$\text{Surface_String_Likelihood} = P(w_0 \dots w_n). \quad (5.4)$$

In CFGs, the agent receives a reward of +1 for any grammatical sequence and a reward of -1 (the default for any action) for any ungrammatical sequence. In PCFGs, the reward for sequence $w_0 \dots w_n$ corresponds to $P(w_0 \dots w_n)$, i.e. the probability of the sequence according to the human corpus. In HMMs, $P(w_0 \dots w_n)$ corresponds to the forward probability—derived from the Forward algorithm—the probability of observing the generated sequence in the data. In Bayesian networks, we reward sequence $w_0 \dots w_n$ by computing $P(w_0 \dots w_n | e)$, where e represents observed variable-value pairs (also referred to as 'evidence'). We only use one of these four ways to assign rewards at a time and then compare the performance of the learning agent when informed by any particular graphical model. The situation is shown graphically in Figure 5.7.

5.4.1.3 Dimension 3: Balancing Alignment and Variation

The third dimension of the reward function aims to balance the proportion of alignment and variation in a natural and human-like fashion such as discussed in

5. GRAPHICAL MODELS FOR SURFACE REALISATION

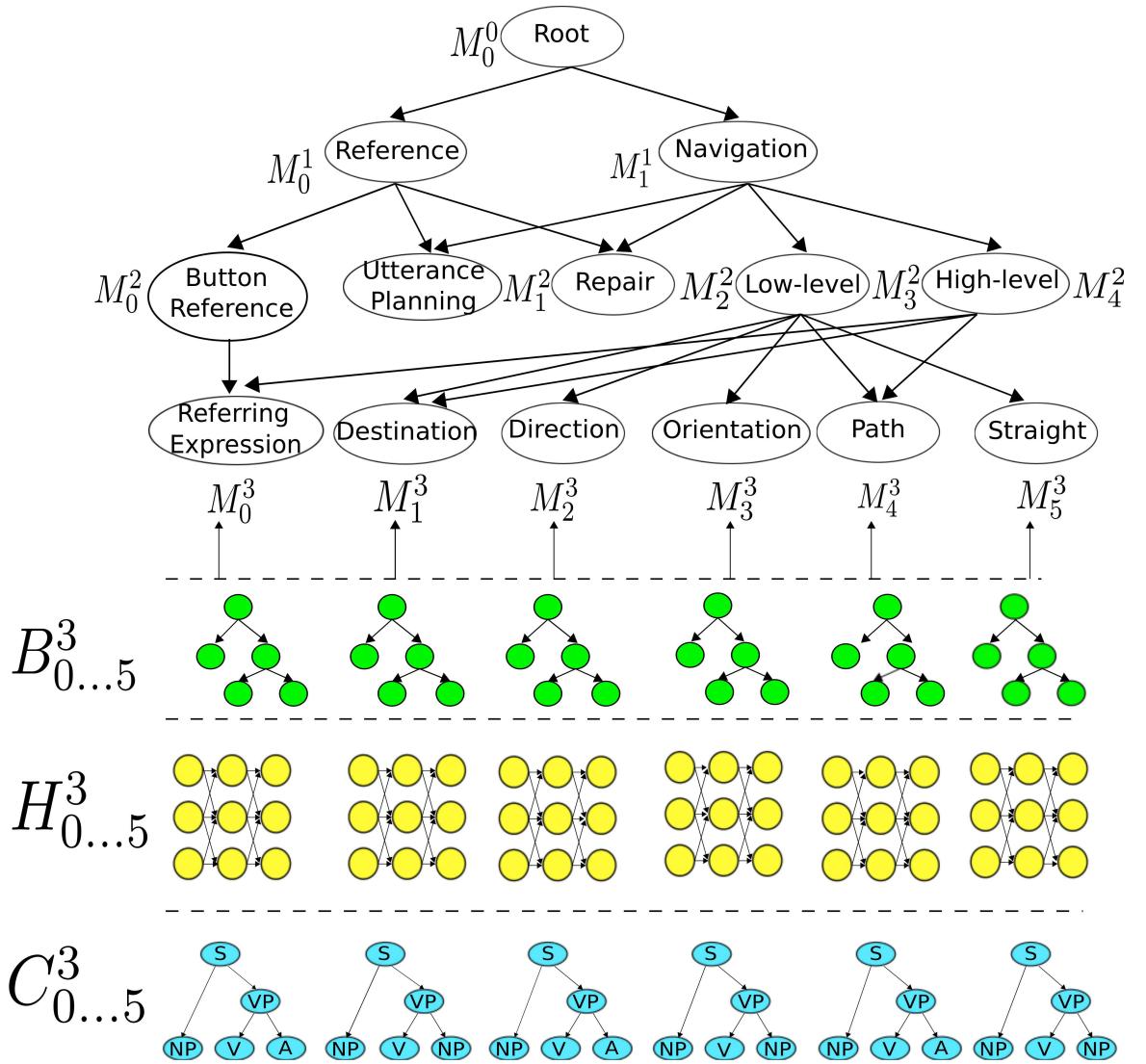


Figure 5.7: The hierarchy of learning agents with alternative graphical models shown in the bottom. They inform the surface realisation agents $M_{0...5}^3$ by providing feedback for the surface realisation choices in the form of rewards. Note that only one type of graphical model is used at a time.

5. Experimental Setting

Section 5.2. To this end, we would like the agent to generate utterances so that the *constituent alignment score* for each utterance is as close to 0.5 as possible. To achieve this, we assign each generated utterance a probabilistic reward sampled from a Gaussian distribution. In probability theory this has a probability density function defined as

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (5.5)$$

where μ refers to the mean and σ^2 refers to the variance. The right-hand side of Equation 5.5 is also commonly denoted as $\mathcal{N}(x|\mu, \sigma^2)$ so that the probability density function that we use for the sampling of rewards can be defined as:

$$P(CAS) \approx \mathcal{N}(CAS|\mu, \sigma^2), \quad (5.6)$$

where in our case we used a mean of $\mu = 0.5$, a variance of $\sigma = 0.2$. A *CAS* score in the range $[0 \dots 1]$ indicates the proportion of alignment and variation. This function is shown in Figure 5.8, where the x -axis of the function corresponds to different *CAS* values (with increments of 0.001) and the y -axis corresponds to a probability that the agent receives as a reward according to the *CAS* of its previously generated utterance. For a *CAS* of exactly 0.5, the agent would receive a reward of nearly 0.89.

5.4.1.4 Bringing All Dimensions Together

For the final experiments, we can bring all dimensions of the reward function together by accumulating them whenever more than one reward applies at a time. For example, at the end of an utterance, usually the reward for the *Performance* of the utterance will apply, the reward for the *Surface_String_Likelihood* and the reward for $P(CAS)$. Accordingly, the reward for the utterance can be computed as

$$Reward = Performance + Surface_String_Likelihood + P(CAS). \quad (5.7)$$

For all dimensions and agents, a reward of -1 is assigned for every action in the hierarchy so as to prevent the agent from choosing actions multiple times

5. GRAPHICAL MODELS FOR SURFACE REALISATION

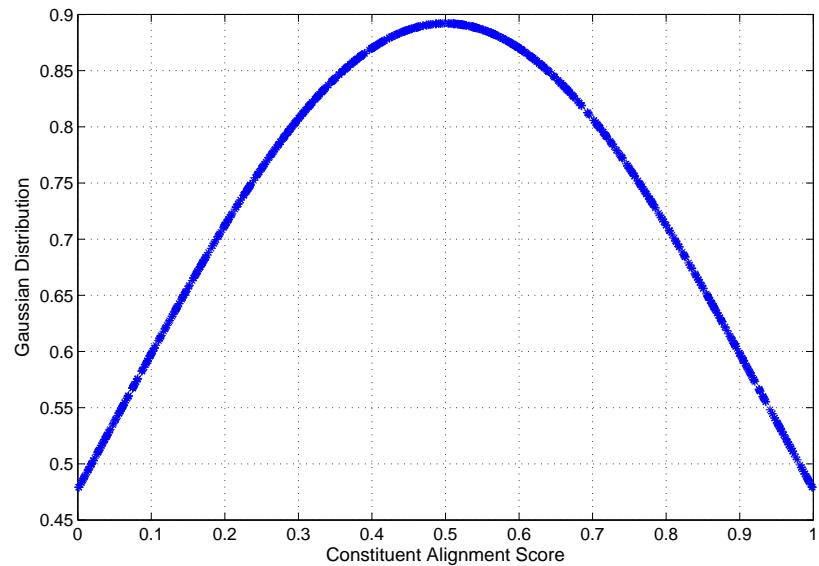


Figure 5.8: The probability density function from which rewards are sampled to inform the agent's learning process with respect to optimising the proportion of alignment and variation. The x-axis corresponds to different *constituent alignment scores* and the y-axis corresponds to probabilities assigned as rewards.

5. Experimental Results

and entering into loops. For example, it could happen that an agent chooses an action repeatedly that has yielded a positive reward in the past (such as choosing a surface realisation for the verb), even though it does not change the state of the environment anymore and instead fails to take other relevant actions (such as choosing a surface realisation for the direction). A small negative reward for repeated actions that do not change the state of the environment can therefore prevent such loops.

Using this three-dimensional reward function, we are now in a position to re-train our learning agent and evaluate its resulting policy from several perspectives. It is expected to be able to generate consistent and successful sequences of instructions by taking the user’s reactions and actions into account. It should adapt flexibly to new and unseen circumstances and display sophisticated troubleshooting strategies whenever the user gets lost or confused. In addition, it should make surface realisation choices that have a high likelihood but at the same time show variation. Finally, there are important trade-offs between the tasks of content selection, utterance planning and surface realisation which should be addressed and optimised through the joint treatment using the proposed unified hierarchical architecture.

5.5 Experimental Results

5.5.1 Simulation-Based Results

In this section, we will compare the policies that the learning agent learnt when informed by different graphical models and their performance in the simulated environment. We will see that both Bayesian Networks and HMMs lead to good policies, while PCFGs and CFGs lead to suboptimal behaviour.

5.5.1.1 Comparison of Policies Using Graphical Models

A range of different graphical models were used to inform the agent’s surface realisation decisions during training. The agent’s overall challenge in the experiments, though, was to learn an optimal policy that balances the trade-offs of content se-

5. GRAPHICAL MODELS FOR SURFACE REALISATION

lection, utterance planning and surface realisation in a maximally unified and adaptive fashion. In the following we will compare five resulting policies:

1. Jointly learnt with Bayesian Networks (*Joint with BNs*)
2. Jointly learnt with Hidden Markov Models (*Joint with HMMs*)
3. Jointly learnt with probabilistic context-free grammars (*Joint with PCFGs*)
4. Jointly learnt with context-free grammars (*Joint with CFGs*)
5. Isolated Optimisation (*Isolated*)

The isolated optimisation serves as a baseline to the joint optimisation with any graphical model. Its surface realisation behaviour is based on Bayesian Networks as the most context-aware of all models. The policy optimised jointly with CFGs serves as another baseline to the optimisation with graphical models. The first four policies (using a joint optimisation) all learn the same content selection and utterance planning behaviour. They differ with respect to surface realisation, though. In terms of *content selection*, all policies prefer a high-level navigation strategy (for more efficient instruction giving) and switch to low-level or mixed when the user gets confused. They prefer identifying a referent by its colour. Alternatively, they identify it by a spatial relation, a distractor with a discriminating colour or a landmark (in this order of preference). With respect to *utterance planning*, the jointly optimised policies prefer presenting instructions in a one-by-one fashion for three or more instructions and jointly otherwise. More than three instructions are organised by the help of temporal markers. The isolated policy differs from the others as already described in Section 3.6, Chapter 3. It is strongly guided by efficiency considerations preferring mainly high-level navigation, joint presentations of instructions and identification of referents using all available attributes. The reason for this non-optimal behaviour is again the limited knowledge of the actions of other components and their effects.

All policies differ in the *surface realisation* decisions they learn. The policies using Bayesian Networks and HMMs learn to balance the trade-offs of variation and alignment while still acting in accordance with the language model and the user's information need. They get closest to the desirata specified for the reward function in Section 5.4.1. The policy learnt using PCFGs acts greedily on behalf of the language model and always chooses the most likely surface variant without

5. Experimental Results

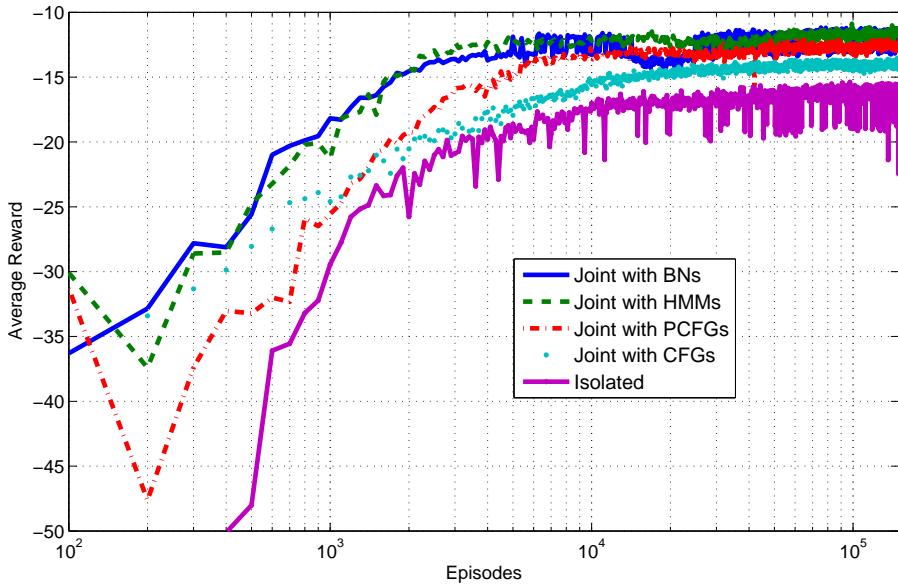


Figure 5.9: Results in terms of average rewards for the joint learning framework including graphical models for surface realisation. In comparison to the plots in Figure 3.11 in Chapter 3, we can see an increase in performance due to the integration of graphical models. Bayesian Networks and Hidden Markov Models achieve the best performance, followed by PCFGs and CFGs.

variation. Apparently, the rewards provided by a maximal *CAS* were overwritten here by the probabilities of surface form variants that seems to have had a bigger weight in learning. Similarly, the policy learnt using CFGs generates grammatical forms randomly without taking any other trade-off into account.

5.5.1.2 Effects of Graphical Models for the Joint Optimisation

Figure 5.9 plots the performance of policies in terms of average rewards (averaged over 10 runs) over 150 thousand episodes. Two effects are visible. First, we see our results from previous chapters confirmed that a joint optimisation of NLG subtasks does indeed lead to an increase in performance in contrast to an isolated optimisation (i.e. jointly optimised policies outperform the purely isolated policy using any graphical model or CFG). All jointly optimised policies (regardless of the graphical model) perform better than the purely isolated policy.

5. GRAPHICAL MODELS FOR SURFACE REALISATION

An absolute comparison of the average rewards of the last 1000 training episodes shows that the (best) jointly learnt policy (using Bayesian Networks) outperforms the isolated policy by 34%, which is significant at $p < 0.0001$ (using a two-tailed paired t-test) with a high effect size r of 0.96.

As a second effect we can see that HMMs reach nearly the same performance as Bayesian Networks. This may seem surprising initially given that Bayesian Networks are more context-aware than HMMs. The reason is that the hierarchical learning agent will discover any non-optimal behaviour that is caused by the HMMs' occasional lack of context-awareness (due to the independence assumption they make) and learn to balance this drawback by learning a more comprehensive policy itself. For the Bayesian Networks this is not necessary, since their random variables and dependencies with other variables allow them to adapt to different contexts.

In order to test this hypothesis, we isolated all graphical models and the CFGs from the learning agent (to prevent communication between them) and re-trained the policies. The isolation is done by allowing the learning agent to not have access to the decisions made by the graphical models, so that decision making by both components is entirely isolated. Note that the content selection and utterance planning behaviour is still optimised jointly, just the graphical models (i.e. surface realisation) are independent. The resulting policies for all graphical models (without communication with the learning agent) are shown in Figure 5.10. We can see that while the agent using Bayesian Networks learns the same policy as before (as do the agents with CFGs and PCFGs), the performance of the agent using HMMs deteriorates drastically. It now achieves an average reward of only -12.12 which is 5% worse than before and only slightly better than the PCFGs. This drop in performance is significant at $p < 0.0001$ with a high effect size $r = 0.79$. These results seem to suggest that while Bayesian Networks are the best performing model in isolation due to their higher context-awareness, HMMs represent a scalable and easier-to-train alternative when applied in a joint learning framework.

5. Experimental Results

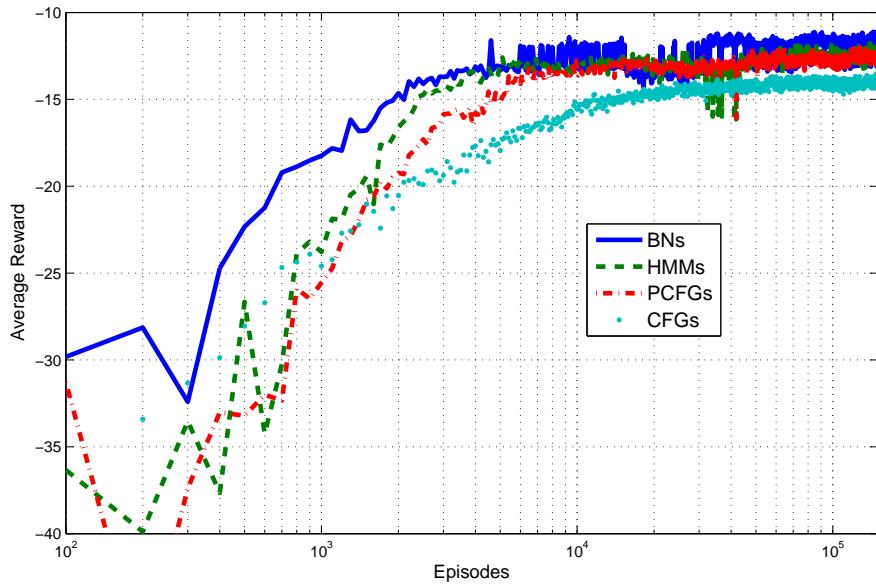


Figure 5.10: Results in terms of average rewards that different graphical models achieve in isolation of the joint learning framework. We can see that Bayesian Networks reach the same performance within and without the joint framework, but the performance of Hidden Markov Models deteriorates when considered in isolation. PCFGs and CFGs reach the same (suboptimal) performance as before.

5. GRAPHICAL MODELS FOR SURFACE REALISATION

5.5.2 Similarity with Human Authors

Our main objective in balancing the proportion of alignment and variation in system utterances was to make them more natural and human-like. To evaluate the instructions generated by each policy (using different graphical models) according to this objective, we compare them using Precision-Recall based on the F-Measure score [Jurafsky and Martin, 2009] and similarity based on the Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951], which was applied to dialogue by Cuayáhuitl et al. [2005]. The Precision-Recall metric evaluates how well a trained model can make predictions about the test data. *Precision* is defined by

$$\text{Precision} = \frac{\text{number of correct surface forms}}{\text{number of all surface forms}}, \quad (5.8)$$

and *Recall* is defined by

$$\text{Recall} = \frac{\text{number of correct surface form}}{\text{total number of surface forms in the data set}}. \quad (5.9)$$

Based on these two metrics, we can then compute the F-score as

$$\text{F-score} = \frac{2PR}{P+R}, \quad (5.10)$$

where P refers to Precision and R refers to Recall. Note that the notion of ‘correct’ in surface realisation is very strict and therefore not always applicable. We will therefore give a higher importance to the similarity results based on the Kullback-Leibler divergence. The KL-Divergence score computes the normalised difference between two probability distributions P and Q based on unigram language models. This is done in two steps. First, we compute the smoothed probability distributions for two data sets, so that P represents the distribution over the human data, and can therefore be seen as a gold standard, and Q represents the distribution over the generated data. Using both, we can compute the symmetric distance according to

$$D(P, Q) = \frac{D_{KL}(P||Q) + D_{KL}(Q||P)}{2}, \quad (5.11)$$

5. Experimental Results

Compared Instructions	F-Score (Precision-Recall)	KL-Divergence (Similarity)
Real1 - Real2	0.58	1.77
Real - <i>Joint with BNs</i>	0.38	2.83
Real - <i>Joint with HMMs</i>	0.40	2.80
Real - <i>Joint with PCFGs</i>	0.49	4.34
Real - <i>Joint with CFGs</i>	0.0	10.06

Table 5.2: Comparison of surface forms generated with graphical models and CFGs against human-authored instructions in terms of Precision-Recall (the higher the better) and KL-Divergence (the lower the better). The first row compares two human data sets and the remaining rows compare both together (Real1 + Real2) against a data set generated with BNs, HMMs, PCFGs and CFGs.

where D_{KL} is the Kullback-Leibler Divergence between P and Q :

$$D_{KL}(P||Q) = \sum_x p_x \log_2 \frac{p_x}{q_x}. \quad (5.12)$$

The lower the KL-Divergence between two data sets, the more similar they are. Conversely, the higher the KL-Divergence between two data sets, the less similar the data are. Please see [Pietquin and Hastie \[to appear\]](#) for an overview of dialogue simulation evaluation metrics.

As a gold standard we use the human-authored instructions from the GIVE corpus and split them in half. This results in two data sets ('Real1' and 'Real2') whose F-score and similarity will serve as the upper bound for the comparison with graphical models. All comparisons are based on surface strings either drawn directly from the human corpus (for the gold standard) or generated using our learnt policies (for the graphical models). Table 5.2 shows the results of comparing each of Bayesian Networks, HMMs, PCFGs and baseline CFGs against the conjunction of the two human data sets ('Real'). We can observe that while *HRL with PCFGs* receives higher F-scores than the other graphical models, both *HRL with BNs* and *HRL with HMMs* are more similar to the human data. We can also see that Bayesian Networks and HMMs are again very similar in their performance confirming our simulation-based results from the previous section. Presumably, the fact that PCFGs achieve the highest F-score is due to

5. GRAPHICAL MODELS FOR SURFACE REALISATION

the alignment–variation trade-off. Since the agent learns a greedy policy using PCFGs that always exploits the most likely variant (but showing no variation), the F-score is boosted, but the KL-Divergence (i.e., the similarity with human authors) is low. Ultimately, it is thus desirable for the Bayesian Networks and HMMs to have a low KL-Divergence score even at the cost of a lower F-score. It enhances the naturalness and human-likeness of the generated instructions. *HRL with CFGs*, which generates grammatical surface forms with random variation, is unsurprisingly rated worst by both metrics.

5.6 Conclusion

This chapter has focused on enhancing the quality of surface realisation as the last component to be optimised in our hierarchical joint learning framework. At the end of Chapter 3, we observed several problems with surface realisation that led to unnatural instructions that were sometimes hard to comprehend. The reason for this was the random choice policy that the agent had learnt due to the absence of a language model to inform the agent’s decision making progress. In addition to the probabilistic information provided by a language model, we noticed that humans vary and align their surface realisation choices in a proportion of roughly 50% – 50%. While this balance does not seem to impact task success or even user satisfaction, it may enhance the naturalness and human-likeness of system-generated discourse. We decided to tackle both problems by representing surface realisation choices systematically as generation spaces that could provide feedback to the agent’s learning process.

Several graphical models have been suggested as representations of generation spaces in the recent literature. We decided to use probabilistic context-free grammars, Hidden Markov Models and Bayesian Networks, integrate them into our joint learning framework and compare their performance in practice. To this end, they were trained on the human corpus data and utilised to provide feedback to the agent’s surface realisation decisions during learning. As a result, we found that Bayesian Networks and Hidden Markov Models achieve the best performance both in simulation and according to the similarity of their generated instructions to human-authored instructions. While the Bayesian Networks,

5. Conclusion

Conversant	Utterance
USR	<i>starts game: ‘How do I get to the trophy?’</i>
SYS	Go until the end of the hall and turn to the right.
USR	<i>hesitates</i>
SYS	Turn 90 degrees right, then go straight.
USR	<i>executes instructions</i>
SYS	Now go until the end of the corridor and turn left.
USR	<i>executes instructions</i>
SYS	Very good. Now walk to the red button left of the blue button.
USR	<i>executes instructions</i>
SYS	Click the red button.
USR	<i>performs desired action: presses red button</i>
SYS	Very good.

Table 5.3: Updated sample dialogue (from Chapter 3, Table 3.5) using the policy optimised jointly including Bayesian Networks for surface realisation. In comparison to the surface realisation choices of the earlier dialogue, the Bayesian Networks here contribute to more natural and human-like language. The content selection and utterance planning strategies are largely unchanged.

5. GRAPHICAL MODELS FOR SURFACE REALISATION

however, reach the same performance within and without the joint learning framework, the performance of HMMs drops drastically when used in isolation [[Dethlefs and Cuayáhuitl, 2012](#)]. This is due to the higher context-awareness of Bayesian Networks. Their random variables allow them to adapt to linguistic as well as situational information. We can conclude from these results that while Bayesian Networks are the most successful model in isolation, HMMs offer a cheap and scalable alternative when used in conjunction with (hierarchical) reinforcement learning. This is especially attractive given the scalability problems that Bayesian Networks often encounter when applied to large problems.

At the moment, our generation space models are language-dependent because they were trained on English corpus data. However, the method suggested in this chapter by itself is language-independent, so that generation spaces can be trained for any language for which data is available.

Even though the simulation- and automatic metric-based evaluations in this and the previous chapters have provided support for our joint learning framework (see Table 5.3 for the example dialogue from Table 3.5 in Chapter 3 with updated surface realisation choices), an open question so far is whether humans would actually perceive the alleged improvements developed in this thesis. The next chapter will present a human evaluation study to investigate this question. As a support to surface realisation, we will use Bayesian Networks since they were the best performing graphical model in this chapter.

Chapter 6

Evaluation

6.1 Introduction

A jointly optimised NLG policy considers decisions of content selection, utterance planning and surface realisation in an interrelated fashion and thus corresponds better to their interdependent nature than a policy optimised in isolation. Consequently, it can lead to more successful interactions with fewer problems that are more positively perceived by users. So far all evidence presented in favour of this hypothesis was based on simulation (Chapters 3 and 4), preliminary user ratings (Chapter 4) and automatic metrics (Chapter 5). An open question remains whether the same effects hold for interactions with real human users. In this chapter, we will present a task-based user study to evaluate the joint hierarchical reinforcement learning approach we have developed so far. This includes constraints to represent prior knowledge and graphical models to support natural surface realisation choices. The evaluation presented will have two parts. First, we will present an NLG system that is based on the GIVE virtual environment. Here we will test the designed joint learning framework in the domain for which it was developed. Second, we will transfer our approach to a new, but related, domain and test the domain transferability of the joint learning framework. To this end, we will use a mobile application that assists users in navigating through a real university building by generating instructions along the way—while users navigate—for each decision point they encounter.

For the evaluation of the GIVE scenario, an important assumption is that the differences in terms of content selection, utterance planning and surface realisation that we observed for the joint and isolated optimisation in Chapters 3, 4 and 5 will also be perceived by human users. Content selection and utterance planning choices should lead to certain objective differences, for example, in the number of user turns or user false actions, where a higher number can indicate an increased cognitive load. Similarly, binary and graded task success are expected

6. EVALUATION

to be higher for more optimal content selection or utterance planning choices. The differences should also be visible in terms of user satisfaction, though, where a higher user satisfaction is expected depending on the smoothness of an interaction. With respect to surface realisation, we expect that users will perceive the jointly optimised system (using graphical models) as more natural and less robotic than its isolated counterpart. The ultimate hypothesis for the GIVE evaluation scenario will be that the joint optimisation framework leads to more successful and efficient interactions that are better perceived by human users. The null hypothesis we wish to reject is that there are no noticeable differences between the joint and the isolated policy. Note that the evaluation also serves as a measure of the generalisability of the learnt policies which are tested in worlds that are different from the ones in which they were trained.

As a second part of the evaluation we would like to transfer our approach to a new, but related, domain in order to test the ease with which such a transfer can be done and the performance that can be achieved. As a new domain, we will apply our method to a real situated indoor navigation scenario in a university building that is generally considered complex to navigate. In this scenario, users will navigate to a set of locations in the building using a mobile application. It guides them by generating instructions for each decision point that they encounter. The system was initially developed for the generation of in-advance route instructions [Cuayáhuitl, 2011; Cuayáhuitl and Dethlefs, 2011a,b] and was newly equipped to also support in-situ interactions. Note that all route planning and content selection decisions will be made by the route planner and the dialogue manager of the system, respectively, the NLG component will be restricted to surface realisation and utterance planning. As a result of applying our method to a navigation system for situated interaction, we expect high binary and graded task success scores of the system. In addition, it would be desirable to obtain high user satisfaction scores indicating appropriate communicative skills of the system, appropriate guidance and interaction pace. The hypothesis for this part of the evaluation will be that the method proposed in this thesis can be transferred to new related domains with limited effort and contribute to high task success and user satisfaction scores. We will in this context also compare the in-situ wayfinding scenario to an in-advance wayfinding scenario.

6. Experimental Setting for Navigation in Virtual Environments

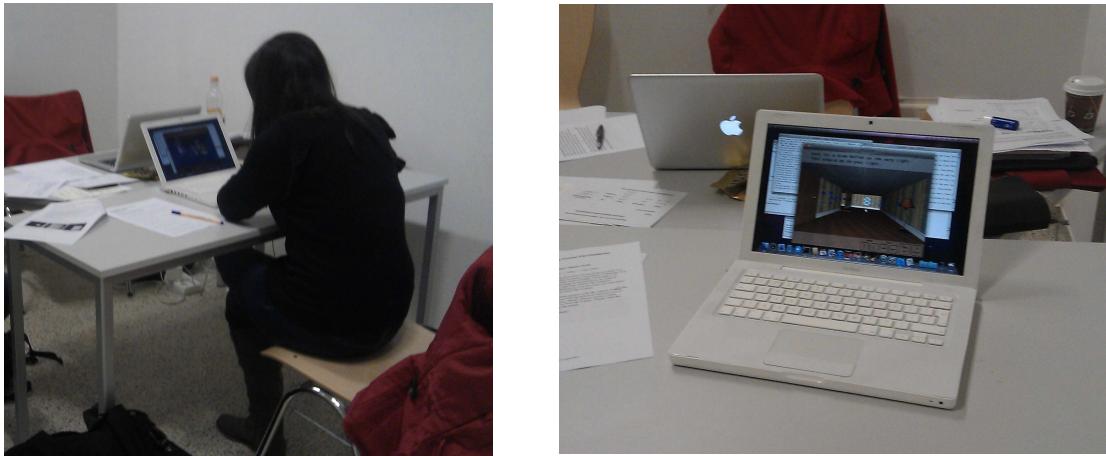


Figure 6.1: Photos of the experimental setting for the GIVE evaluation. Left: a participant is playing. Right: the computer that participants played on.

6.2 Experimental Setting for Navigation in Virtual Environments

After having developed a general learning framework for an NLG system that generates instructions for situated interaction in the GIVE task, we will evaluate the system in three different virtual worlds with human participants. The evaluation will involve objective and subjective metrics and be carried out in a lab environment. Figure 6.1 shows photos of the experimental setup.

6.2.1 Experimental Methodology

We use objective and subjective metrics based on the PARADISE framework [Walker et al., 1997] for evaluating dialogue systems in order to evaluate our systems for the GIVE task. The metrics are similar to the ones we used to induce the learning agent’s reward function in Section 3.5.2. Table 6.1 gives an overview of the objective metrics that we use to evaluate the two system versions, jointly optimised and optimised in isolation. Under the category *interaction efficiency*, we find metrics such as the time that an interaction took, the number of system

6. EVALUATION

turns and system words, and the number of user turns (we count as user turns help requests or hesitations that last longer than a prespecified threshold of 4 seconds). Under the *interaction quality* category, we count the number of user help requests and user hesitations (the sum of which correspond to the ‘user turns’ metrics under *interaction efficiency*), the number of false user actions overall, the number of false user navigation actions and the number of false user manipulation actions (i.e., false button presses). The false user actions overall metric corresponds to the sum of false navigation and manipulation actions. Finally, under the category *task success*, we distinguish binary task success (won or lost) from 3-valued and 4-valued graded task success (GTS) which penalises task difficulty in different ways. The 3-valued graded task success metric can be expressed as follows, where FTL stands for ‘finding the target location’. We assume that every user turn (hesitation or help request) indicates a problem.

$$3\text{-valued GTS} = \begin{cases} 1 & \text{for FTL without problems} \\ 1/2 & \text{for FTL with small problems} \\ 0 & \text{otherwise.} \end{cases}$$

We assign the value of 1/2 (small problems) for more than five user turns and the value of 0 (otherwise) for more than ten user turns. Similarly, the 4-valued graded task success metric can be expressed as follows.

$$4\text{-valued GTS} = \begin{cases} 1 & \text{for FTL without problems} \\ 2/3 & \text{for FTL with small problems} \\ 1/3 & \text{for FTL with severe problems} \\ 0 & \text{other.} \end{cases}$$

The value of small problems is again assigned for more than five user turns, the value of severe problems is assigned for more than ten user turns and the value of 0 is assigned when the user loses the game.

The objective metrics were designed based on the PARADISE metrics, but tailored specifically to our scenario and so as to measure the success of instruction generation in a situated interaction scenario. All objective metrics were computed automatically and saved into log files during the interaction.

6. Experimental Setting for Navigation in Virtual Environments

Table 6.1: Objective metrics for the GIVE evaluation that were logged during interactions. These metrics are very similar to those used to induce a reward function from data in Section 3.5.2.3.

Objective Metric	Description
Interaction Efficiency	
(O1) Elapsed time	How long was the interaction?
(O2) System turns	How many system turns?
(O3) System words	How many system words overall?
(O4) System words per turn	How many system words per turn?
(O5) User turns	How many user turns (help requests, hesitations)?
Interaction Quality	
(O6) User help requests	How many user help requests?
(O7) User hesitations	How many user hesitations?
(O8) User false actions (all)	How many false user actions overall?
(O9) User false actions (navigation)	How many false user navigations?
(O10) User false actions (manipulation)	How many false user manipulations?
Task Success	
(O11) Binary Task Success	Was the game won or lost (cancelled counts as lost)?
(O12) 3-Graded Task Success	Was the game won without problems, won with small problems, or other?
(O13) 4-Graded Task Success	Was the game won without problems, won with small problems, won with severe problems or lost?

6. EVALUATION

Table 6.2: Subjective metrics for the GIVE evaluation and the questions that were asked in order to obtain them. These metrics are very similar to those we used to induce a reward function from data in Section 3.5.2.3.

Subjective Metric	Question
(Q1) Easy to Understand	Was it easy to understand the system?
(Q2) Task Easy	Was it easy to find the trophy?
(Q3) Interaction Pace	Was the speed of the interaction okay?
(Q4) What to Do	Did you know at each moment what to do?
(Q5) Expected Behaviour	Did the system work as you expected it to?
(Q6) Future Use	Would you use this system in the future?
(Q7) Appropriate Help	Did the system give you appropriate help when you needed some?
(Q8) Enjoy Game	Did you enjoy the game?
(Q9) Recommend to Friend	Would you recommend the system to a friend?
(Q10) Language Robotic	Was the language of the system robotic?

Table 6.2 shows the subjective metrics we use to evaluate the user satisfaction of our two systems. While questions Q1-Q6 are taken nearly directly from PARADISE, questions Q7-Q10 were included to test some specifics of our situated NLG scenario. These metrics were obtained through questionnaires that participants were asked to fill after each game they played.

6.2.2 Experimental Setup

We compared two systems for the GIVE task in a human evaluation study involving 20 participants, 80% (16 out of 20) female and 20% (4 out of 20) male, with an age average of 24.5. They took part to fulfill partial course requirements or against payment. The two systems to be compared generated instructions for the GIVE task in three different worlds, which were chosen to be different from the training worlds, in order to assess the generalisability and flexibility of our learnt policies. One system used a jointly optimised policy, the other system used a policy that was optimised in isolation. Participants were asked to play three games, one in each world in a randomised order (from a uniform distribution). They were chosen so as to ensure that each participant played with at least one jointly optimised system and one system optimised in isolation. Apart from this

6. Experimental Setting for Navigation in Virtual Environments

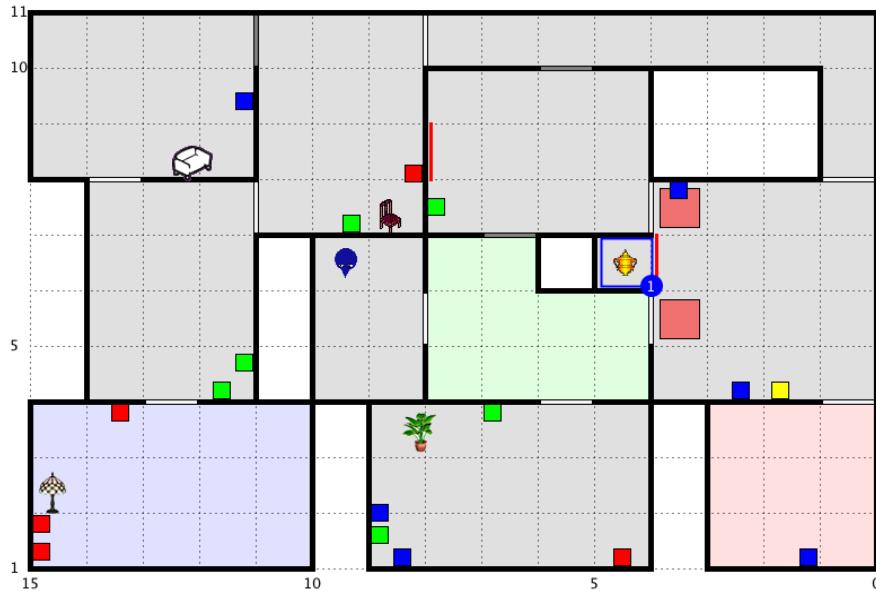


Figure 6.2: Illustration of training world 1. The focus here is on navigation of a low to medium level of difficulty and referring expressions that require a limited amount of disambiguation.

condition, systems were chosen randomly from a uniform distribution. During the games, all information relevant to estimate the objective interaction metrics was saved into log files. Following each game, participants were asked to fill a questionnaire on a set of subjective metrics which we used to assess their overall user satisfaction using a 1 to 5 Likert scale on which 5 represents the best score and 1 represents the worst. In the next section, we introduce the worlds used for training and evaluation and compare their characteristics and challenges. The training worlds were taken from the GIVE development environment¹, the evaluation worlds were used from the official GIVE challenge 2.5² from 2011 [Striegnitz et al., 2011].

¹<http://www.give-challenge.org/research/page.php?id=software>

²<http://www.give-challenge.org/research/page.php?id=give-2.5-index>

6. EVALUATION

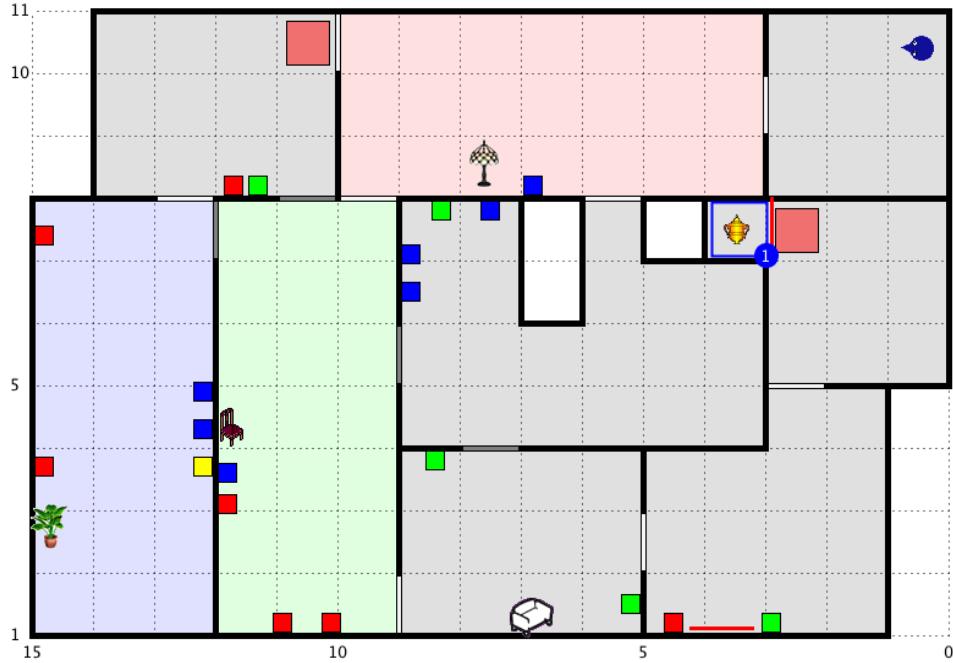


Figure 6.3: Illustration of training world 2. The challenge of this world is represented by its large rooms, which require high-level navigation skills in certain situations (such as crossing a large room or referring to a particular corner).

6.2.2.1 Training Worlds

We used three training worlds in order to design and train our learning agent. Human interaction data from each world was annotated as described in Sections 3.4.2 and 3.4.3 in order to motivate the learning agent’s state and action spaces as described in Appendix A. Apart from this though, the agent needs to explore different spatial settings and configurations including a range of different objects, buttons and spatial arrangements. To this end, we estimated the stochasticity of different spatial arrangements according to their occurrence in the training worlds and used them to inform the situated environment described in Section 3.5.1.

We distinguish three training worlds with different characteristics and associated challenges. World 1 represents a medium level of difficulty with respect to both navigation instructions and referring expressions. Some navigation skills are required to guide the user through a number of medium-sized rooms and past two

6. Experimental Setting for Navigation in Virtual Environments

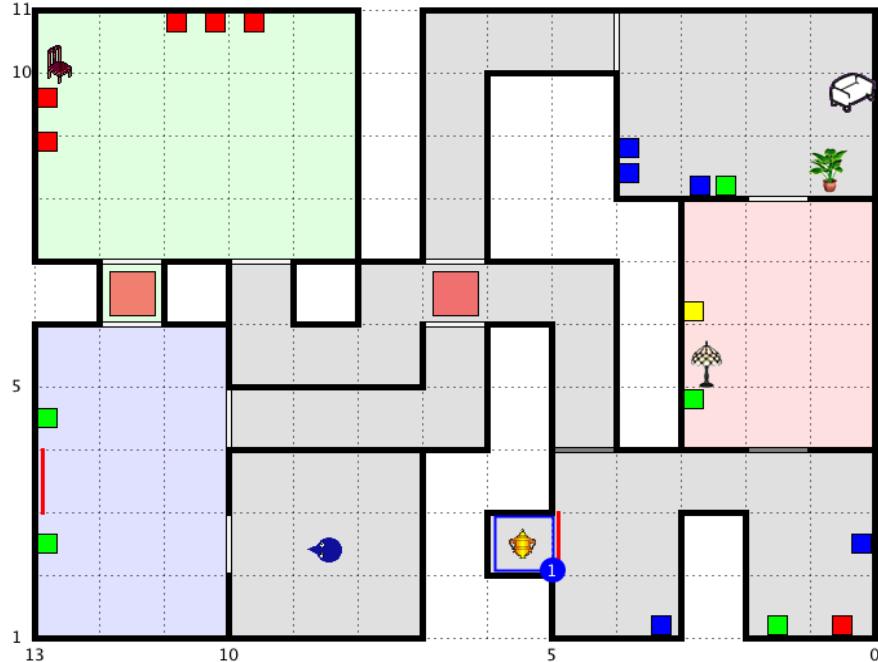


Figure 6.4: Illustration of training world 3. The challenge here lies in guiding the user successfully through a number of small rooms that are connected by a difficult junction in which a user can quickly lose orientation.

red alarm tiles.¹ For successful referring expressions, some basic disambiguation skills are needed; they can in this world mainly rely on using colour and position of referents. Please see Figure 6.2 for an illustrating map of the world. World 2 is similar to World 1 apart from that it contains a number of large rooms, which users can occasionally find hard to navigate. They may thus need extra guidance in certain situations (e.g. when a large room needs to be crossed). Referring expressions can again rely on the basic disambiguation of a small number of competing buttons (for example using their colour or position) with respect to distractors or nearby landmarks. A map of the world is shown in Figure 6.3.

World 3 finally contains a larger number of small rooms than the other two worlds so that users may need extra guidance when trying to find their way through them. In addition, the rooms are connected by a complex junction which

¹When a user steps onto a red tile, an alarm is activated and the game is lost. Red tiles should therefore be avoided.

6. EVALUATION

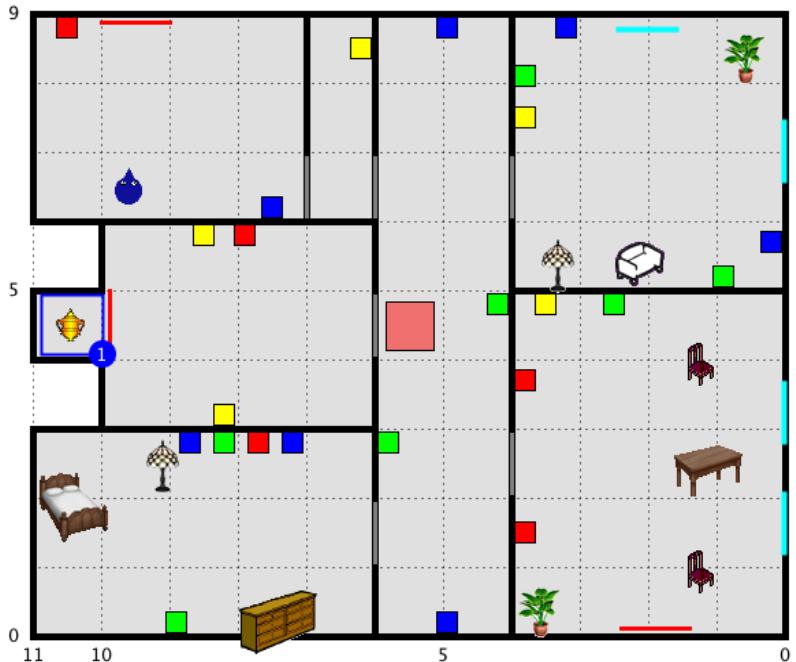


Figure 6.5: Illustration of evaluation world 1. This evaluation world was designed to be similar to the training worlds. It requires navigation with a medium degree of difficulty and limited disambiguation with respect to referring expressions.

users can easily get lost in when they do not receive appropriate instructions. Referring expressions on the other hand represent just a medium level of difficulty again similar to the other two worlds. Please refer to Figure 6.4 for an illustration of the world.

6.2.2.2 Evaluation Worlds

In order to test the generalisability of our learnt policies, we evaluate our GIVE systems in three evaluation worlds which differ from the training worlds along a number of dimensions. Evaluation world 1 is most similar to the training worlds of all three evaluation worlds. Even though it contains more objects and buttons, navigation in the world is not overly complex and referring expressions can again rely on basic disambiguation skills. Nevertheless, due to its larger number of objects, this evaluation world is likely to present a bigger challenge to users than

6. Experimental Setting for Navigation in Virtual Environments

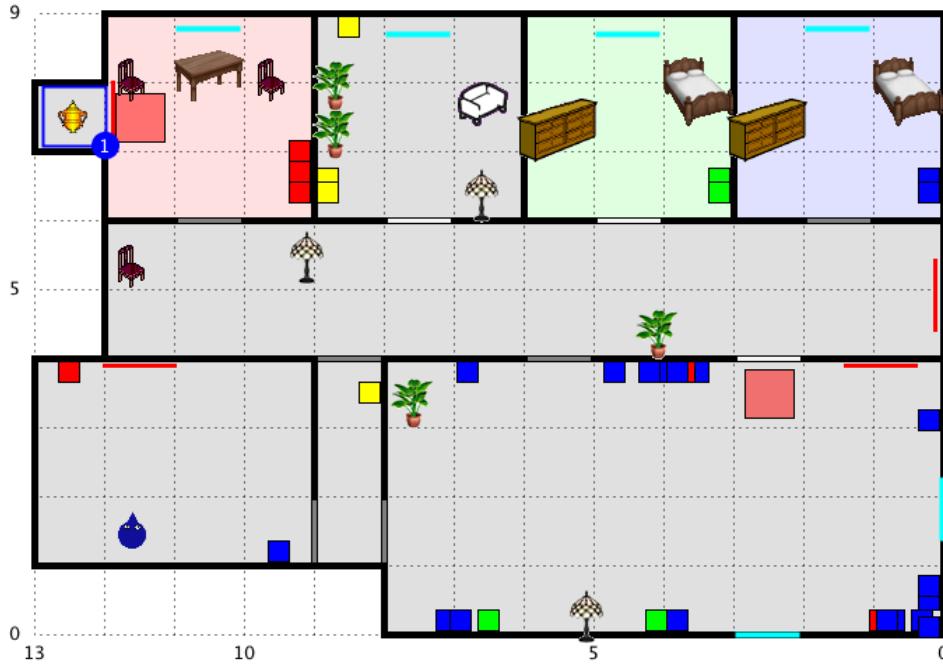


Figure 6.6: Illustration of evaluation world 2. In this world, the focus is on referring expression generation. A large number of same-coloured buttons are located close to each other in different spatial arrangements so that disambiguation becomes a challenge.

the training worlds. A map of the world is shown in Figure 6.5.

Evaluation world 2 has a clear focus on referring expression generation. While navigation is relatively simple—located mainly within one spacious room and along a corridor with several smaller rooms—the world contains a number of difficult referring expression generation situations. These are mainly characterised by multiple buttons of the same colour that are located close to each other in complex spatial arrangements and need to be disambiguated from each other. Examples can be seen in each room of the world shown in Figure 6.6.

Evaluation world 3 in contrast focuses clearly on navigation. While referring expression generation in this world is relatively simple, guiding the user to the intended referent represents a real challenge. Advanced navigation skills are needed in three situations in this world. First, the green corridor of the world quickly turns into a maze in which users can lose orientation and get lost. Second, a

6. EVALUATION

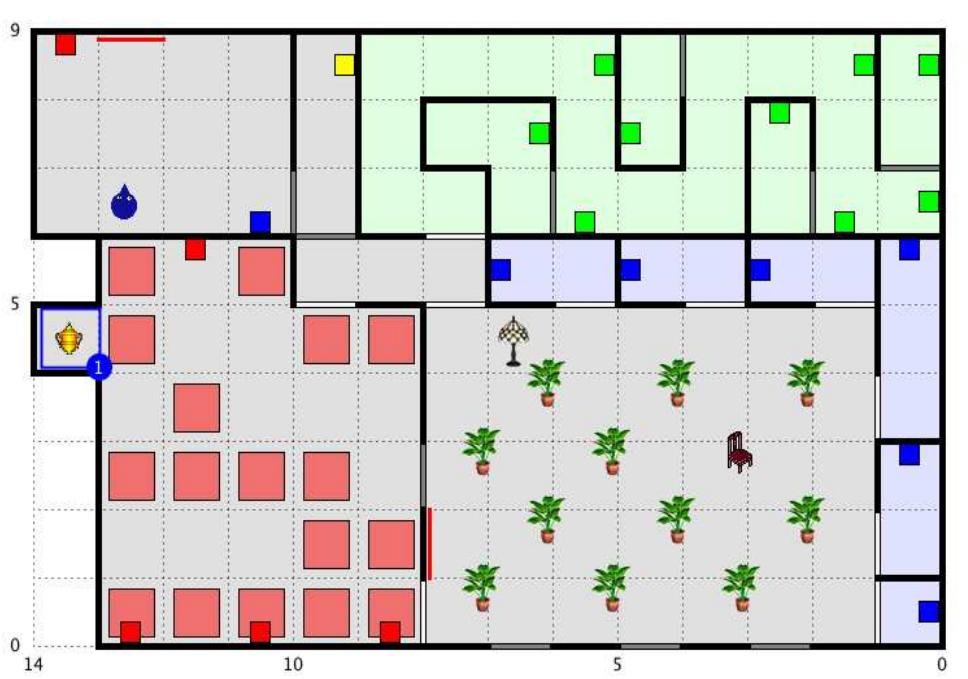


Figure 6.7: Illustration of evaluation world 3. This world requires sophisticated navigation skills in all rooms (the green corridor in which users can quickly lose orientation, the room with the red tiles where any wrong step may cause the alarm to be triggered and the room with the plants where specific references to doors need to be given).

room full of red tiles requires the system to guide the user to three different red buttons on different walls, but without stepping on any of the red alarm tiles. Finally, a room full of plants contains a number of doors that the user needs to go through in a specific order to press a blue button. All three situations require non-trivial navigation skills. Evaluation world 3 is therefore predicted to represent the greatest challenge to both system and user. In the next section, we will present and discuss the results that were obtained for the GIVE task.

6.2.3 Experimental Results

Following the human evaluation study, we analysed the results in order to draw conclusions with respect to the effects that a joint or an isolated optimisation can

6. Experimental Setting for Navigation in Virtual Environments

Table 6.3:

Results for the objective evaluation metrics **per policy** for the joint and the isolated policies. The objective metrics are organised into three groups, *interaction efficiency* (the lower values, the better), *interaction quality* (the lower values, the better) and *task success* (the higher values, the better). Numbers in the second and third column refer to averages and are given together with their standard deviations. The final column shows *p*-values for the comparison obtained with an unpaired two-tailed t-test. All metrics refer to averages over single games.

Objective Metric	Joint	Isolated	<i>p</i> -value
Efficiency			
(O1) Elapsed Time	11:40 ± 5:54	13:19 ± 4:56	0.28
(O2) System turns	314.5 ± 151.3	330.2 ± 80.6	0.7
(O3) System words	3025.3 ± 1522	3887.7 ± 1271	0.04
(O4) System words per turn	9.6 ± 1.3	12.1 ± 1.5	0.0001
(O5) User turns	20.7 ± 14.0	22.2 ± 8.7	0.7
Quality			
(O6) User help requests	2.4 ± 3.0	2.0 ± 1.4	0.6
(O7) User hesitations	18.4 ± 12.6	20.3 ± 8.7	0.5
(O8) User false (all)	21.4 ± 15.5	19.3 ± 6.9	0.6
(O9) User false (navigation)	11.0 ± 7.6	12.8 ± 6.2	0.4
(O10) User false (manipulation)	10.3 ± 10.4	6.5 ± 3.7	0.1
Task Success (TS)			
(O11) Binary TS	0.8 ± 0.4	0.6 ± 0.5	0.1
(O12) 3-Graded TS	0.3 ± 0.3	0.04 ± 0.2	0.008
(O13) 4-Graded TS	0.4 ± 0.3	0.2 ± 0.2	0.009

have on interactions and user satisfaction. Of altogether 20 participants and 60 games, we discarded one participant entirely who later served as an assistant in the evaluation study. This left 56 games for analysis.

6.2.3.1 Objective Metrics

All information relevant to the objective evaluation metrics was saved into log files during the interaction that were analysed subsequent to the human evaluation study. The results for the objective metrics are summarised in Table 6.3.

Effects of Joint and Isolated Optimisation The table compares average results (with their corresponding standard deviations) for the joint and the iso-

6. EVALUATION

lated setting and shows the p -values indicating the significance of the comparison between both settings. We can see that the jointly optimised system performs better than the system that was optimised in isolation according to almost all metrics. It produces shorter interactions using fewer words and turns and causes fewer user turns and hesitations and higher task success. The key findings of the comparison of the joint and the isolated policy in terms of objective interaction metrics can be summarised as follows.

- The isolated policy produces significantly more system words per game (O3) than the joint policy ($p < 0.04$). This difference could be interpreted as a suboptimal content selection policy that was learnt in isolation. When the joint policy is able to achieve an equal (or higher) task success using fewer words, it seems likely that the isolated policy included redundant details at some points.
- The isolated policy produces significantly more system words per turn (O4) than the joint policy ($p < 0.0001$). This difference could again be due to a suboptimal content selection policy on the part of the isolated optimisation, but it could also point to a suboptimal utterance planning strategy. Remember that the cognitive load that is imposed on the user during interaction is increased with the number of system words per turn that the user needs to keep in mind. (Unnecessarily) long utterances can therefore lead to user confusions and affect task success.
- The joint policy achieves better task success than the isolated policy. While the difference in terms of binary task success (O11) only shows a statistical trend ($p < 0.1$), the difference in graded task success (O12 and O13) is significant at $p < 0.0008$ and $p < 0.0009$, respectively. One can therefore say that users interacting with the joint policy encounter fewer problems and experience more smooth and successful interactions. This trend is illustrated by the bar plots in Figure 6.8, which show that the joint policy outperforms the isolated policy according to all task success metrics.

The comparison of the joint and the isolated policy thus seems to suggest that a joint optimisation leads to shorter, more efficient and more successful

6. Experimental Setting for Navigation in Virtual Environments

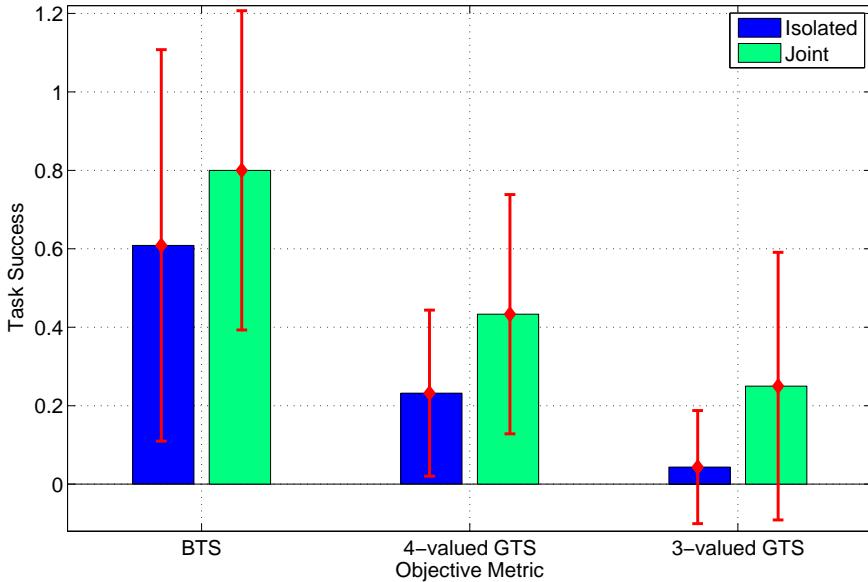


Figure 6.8: Bar plots illustrating the comparison of the joint policy against the isolated policy in terms of binary task success and 4-valued and 3-valued graded task success. The joint policy outperforms the isolated policy according to all metrics indicating that joint interactions are more successful and smooth than their isolated counterparts and encounter fewer problems.

interactions. An exception to the overall trend of the joint policy outperforming the isolated policy is represented by metric O8, the number of false user actions overall, and metric O10, the number of false manipulation actions, i.e. wrong button presses. While users of the joint policy press on average 10.3 (± 10.4) wrong buttons, users of the isolated policy press only 6.5 (± 3.7) wrong buttons on average. The difference is not significant, though. Based on the high standard deviation of the O10 metric of the joint policy, a possible interpretation is that only few users in the joint setting pressed a very high number of wrong buttons, while the majority of users pressed very few (or no) wrong buttons.

Effects of Game Worlds Apart from the differences we observed with respect to the joint or isolated optimisation, there are also a number of differences among the individual game worlds. They are summarised in Table 6.4, where the first three numeric columns show the average results per world (averaged over both

Table 6.4:

Results for the objective metrics **per world** for all evaluation games (across joint and isolated policies) organised into the groups of *interaction efficiency*, *interaction quality* and *task success*. Numbers in columns two to four refer to averages and are given together with their standard deviations. Numbers in the last three columns correspond to the *p*-values indicating the significance between categories obtained with an unpaired two-tailed t-test. Metrics again are per game.

Objective Metric	World1	World2	World3	<i>p-val.</i>	<i>p-val.</i>	<i>p-val.</i>
	W1	W2	W3	W1-W2	W1-W3	W2-W3
Efficiency						
(O1) Elapsed Time	12:03 ± 6:16	9:43 ± 3:17	15:19 ± 5:27	.2	.1	.0004
(O2) System turns	315.7 ± 157.9	241.3 ± 78.0	405.8 ± 142.9	.08	.09	.0001
(O3) System words	3751.5 ± 1758	2675 ± 970.5	3841 ± 1606	.02	.9	.01
(O4) System words per turn	12.0 ± 1.8	11.0 ± 1.8	9.2 ± 1.5	.1	.0001	.001
(O5) User turns	21.5 ± 16.3	16.9 ± 8.6	25.8 ± 12.4	.3	.4	.01
Quality						
(O6) User help requests	1.9 ± 2.3	1.6 ± 2.03	3.0 ± 3.8	.8	.3	.2
(O7) User hesitations	19.6 ± 15.3	15.4 ± 7.8	22.8 ± 10.5	.3	.5	.01
(O8) User false (all)	23.5 ± 17.2	11.2 ± 7.6	27.3 ± 12.3	.008	.5	.0001
(O9) User false (navigation)	12.8 ± 8.3	5.8 ± 4.4	17.0 ± 7.3	.003	.1	.0001
(O10) User false (manipulation)	10.7 ± 12.2	5.4 ± 5.2	10.4 ± 7.7	.1	.9	.02
Task Success (TS)						
(O11) Binary TS	0.9 ± 0.4	0.8 ± 0.4	0.5 ± 0.5	.8	.01	.02
(O12) 3-Graded TS	0.3 ± 0.4	0.2 ± 0.3	0.02 ± 0.1	.6	.01	.02
(O13) 4-Graded TS	0.5 ± 0.3	0.4 ± 0.3	0.2 ± 0.2	0.6	0.002	0.003

6. Experimental Setting for Navigation in Virtual Environments

joint and isolated games) with their standard deviations. The last three columns show the p -values for the comparison. Overall we can see that World2 produces the most successful interactions and World3 produces the least successful interactions. The key findings of the comparison can be summarised as follows.

- World2 allows more efficient and better interactions than World1 using fewer systems words ($p < 0.02$) and causing fewer false user actions ($p < 0.008$) especially of type navigation ($p < 0.003$). Recall from Section 6.2.2.2 that the focus of World2 was on referring expression generation, while in World1 it was on a mixture of navigation and referring expression scenarios with medium difficulty. Based on this, the result could indicate that the learning agent learnt a good and robust referring expression generation policy. This may have allowed the agent to efficiently and unambiguously identify referents leading to fewer system turns and fewer user false actions due to the simplicity of the navigation tasks of World2.
- World1 allows less efficient but more successful interactions than World3 using fewer system words per turn ($p < 0.0001$) and achieving a higher binary task success ($p < 0.01$) and a higher 3-valued ($p < 0.01$) and 4-valued (both $p < 0.002$) graded task success. Recall again from Section 6.2.2.2 that World3 contained a number of difficult navigation challenges, while World1 contained a mixture of navigation and referring expression scenarios. Assuming that the agent learnt a more robust policy for referring expression generation than for navigation, this could explain the difference in terms of efficiency and task success.
- World2 allows more efficient, better and more successful interactions than World3 taking less time ($p < 0.0004$), using fewer system turns ($p < 0.0001$), fewer system words ($p < 0.01$), but more system words per turn ($p < 0.001$) and requiring fewer user turns ($p < 0.01$). Moreover, World2 requires fewer user hesitations than World3 ($p < 0.01$), fewer false user actions overall ($p < 0.0001$), fewer false user navigation ($p < 0.0001$) and manipulation actions ($p < 0.02$). Finally, interactions in World2 achieve a higher binary task success ($p < 0.02$) and a higher 3-valued ($p < 0.02$) and 4-valued

6. EVALUATION

($p < 0.0003$) graded task success. These differences again seem to confirm the impression that the agent learnt a more successful referring expression generation policy, which was important in World2, and a slightly less robust navigation policy, which was important in World3. This difference in policies could then be responsible for the more efficient, better and more successful interactions overall in World2.

While differences exist among individual game worlds, they cannot account for the differences found between the joint and the isolated policy since users played with each world the same number of times. Nevertheless, these results suggest that the learning agent should be trained in relatively complex simulated worlds in order to perform robustly when the policies are deployed. Alternatively, the learning agent should learn online, but this is left as future work.

6.2.3.2 Subjective Metrics

The subjective user ratings that were collected during our comparative system evaluation and that indicate the user satisfaction with each system are analysed in two ways. We first consider all games rated by users and then proceed to consider only the ratings of successful games. This helps to see whether differences exist between these two conditions.

Effects of Joint and Isolated Optimisation Table 6.5 summarises the results for both conditions. The upper half of the table focuses on user satisfaction with the joint and the isolated policy for all games. The lower part of the table focuses on user satisfaction with the joint and isolated policy for the subset of successful games. The last column in the table provides the p -value for the comparison of the first two columns. Overall we can see a clear tendency of users that prefer the joint policy over the isolated policy. The user satisfaction ratings for all games can be summarised as follows.

- Users consistently rate the joint policy better than the isolated policy, even though unfortunately none of the differences is statistically significant.
- The metric ‘Expected Behaviour’ (Q5) receives the highest ratings for both the joint (3.67 ± 1.14) and the isolated (3.52 ± 1.03) policy. In turn, ‘Future

6. Experimental Setting for Navigation in Virtual Environments

Table 6.5:

User satisfaction results for the subjective metrics **per policy** (scores range from 1 to 5 and are the better, the higher the score). The upper part of the table shows the results for all games and the lower part of the table shows the results for the subset of successful games. Numbers refer to averages and are given together with their standard deviations. The last column shows *p*-values for the comparison of systems. Note that for (Q10) ‘Language Robotic’, the results were inversed, so that higher number still indicate better ratings.

Subjective Metric	Joint	Isolated	<i>p</i> -value
All Games			
(Q1) Easy to Understand	3.4 ± 1.0	3.26 ± 1.09	0.6
(Q2) Task Easy	3.01 ± 1.02	3.0 ± 1.16	0.9
(Q3) Interaction Pace	2.9 ± 1.32	2.86 ± 1.45	0.9
(Q4) What to Do	3.43 ± 1.02	2.91 ± 1.08	0.08
(Q5) Expected Behaviour	3.67 ± 1.14	3.52 ± 1.03	0.6
(Q6) Future Use	2.6 ± 0.8	2.56 ± 0.89	0.9
(Q7) Appropriate Help	3.3 ± 1.04	2.87 ± 1.21	0.1
(Q8) Enjoy Game	2.95 ± 1.08	2.56 ± 1.03	0.1
(Q9) Recommend to Friend	3.0 ± 1.16	2.56 ± 1.26	0.1
(Q10) Language Robotic	3.36 ± 0.95	3.21 ± 1.07	0.6
Sum	31.62	29.31	
Successful Games			
(Q1) Easy to Understand	3.58 ± 0.97	3.5 ± 1.09	0.8
(Q2) Task Easy	3.29 ± 0.95	3.28 ± 1.07	1.0
(Q3) Interaction Pace	3.33 ± 1.17	3.42 ± 1.15	0.8
(Q4) What to Do	3.54 ± 1.06	3.14 ± 1.02	0.3
(Q5) Expected Behaviour	3.63 ± 1.2	3.71 ± 0.99	0.8
(Q6) Future Use	2.8 ± 0.8	2.85 ± 0.86	0.7
(Q7) Appropriate Help	3.58 ± 0.92	3.35 ± 1.0	0.5
(Q8) Enjoy Game	3.13 ± 1.11	2.92 ± 0.91	0.6
(Q9) Recommend to Friend	3.17 ± 1.2	3.0 ± 0.87	0.6
(Q10) Language Robotic	3.45 ± 1.02	3.35 ± 1.21	0.8
Sum	33.5	32.52	

6. EVALUATION

Use' (Q6) receives the lowest, $2.6 (\pm 0.8)$ for the joint policy and $2.56 (\pm 0.89)$ for the isolated policy. For the later case, the metrics 'Enjoy Game' (Q8) and 'Recommend to Friend' (Q9) are rated similarly low. Especially the metrics Q8 and Q9 can mean that users of the isolated policy enjoyed their games less than users of the joint policy. The metric 'Future Use' in contrast could also have a different interpretation. Users may not have seen the usefulness of using the game in the future because they are not interested in video games. On a scale of 1 (i.e. 'playing never') to 5 (i.e. 'playing very often'), our participants rated themselves as playing video games between 'rarely' and 'never' (1.78). An alternative interpretation is that users found the pace of the interaction too fast, as indicated by the 'Interaction Pace' (Q3) metric, so that slowing down the interaction pace could lead to higher user satisfaction.

- The metric 'What To Do' (Q4) showed the biggest difference in user ratings between the joint (3.43 ± 1.02) and the isolated (2.91 ± 1.08) system. While it is not (quite) statistically significant, it is the closest among all individual subjective categories. It therefore seems that users found instructions generated by the joint system more easy to interpret and they felt more safely guided through the task.
- Users perceive the language of the joint policy as more natural and less robotic than the language of the isolated policy. The 'Language Robotic' (Q10) metric¹ receives an average rating of $3.36 (\pm 1.09)$ in the joint interactions and an average rating of $3.21 (\pm 1.07)$ in the isolated interactions. While this difference is not statistically significant, it provides evidence that users are at least partially sensitive to the jointly optimised surface realisation of the learning agent. This seems especially likely given that users were focused on solving the task rather than paying attention to the surface forms of the systems and rating their naturalness.

¹Note that in this metric, 5 is again the best score even if the corresponding question to elicit user ratings for the category 'Was the language of the system robotic?' may suggest the opposite. We normalised all values after the evaluation to make 5 consistently refer to the best possible score.

6. Experimental Setting for Navigation in Virtual Environments

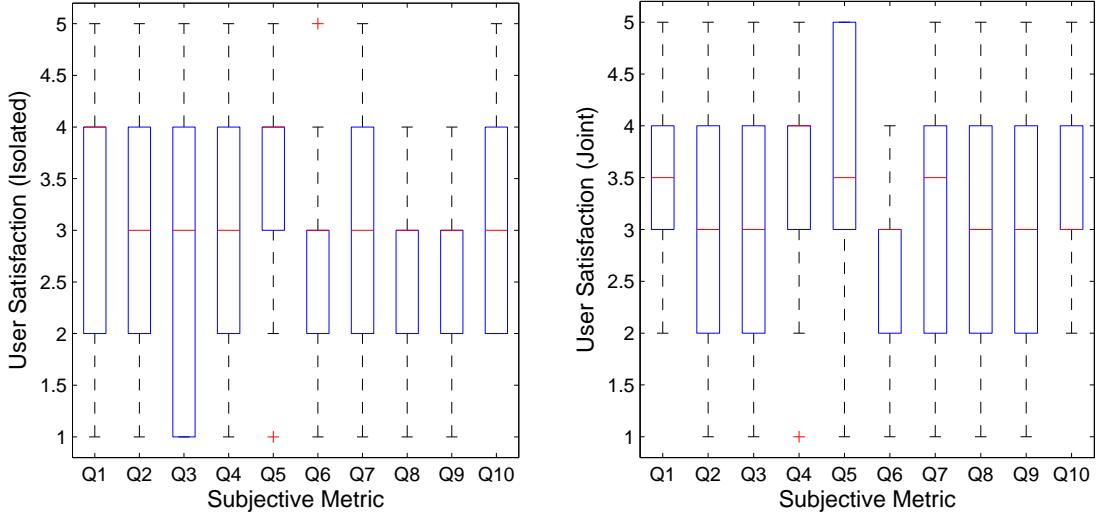


Figure 6.9: Box plots illustrating the user satisfaction ratings for the isolated policy (on the left) and for the joint policy (on the right). The isolated data show a trend towards more negative ratings than the joint policy (metrics Q1, Q4 and Q10) and the joint data show a trend towards more positive ratings than the isolated policy (metrics Q5, Q8 and Q9).

The user satisfaction ratings for only successful games are less clear as shown in the lower part of Table 6.5. On the whole, they show that task success is an important variable in predicting user satisfaction: users are sensitive to their task success (they want to win the trophy) and express this in their system ratings. This observation is also consistent with the task success scores analysed in the previous section. While the joint system had an average binary task success of 0.8, a 3-valued graded task success of 0.3 and a 4-valued graded task success of 0.4, the isolated system only achieved a binary task success of 0.6, a 3-valued graded task success of 0.04 and a 4-valued graded task success of 0.2. The overall difference between ratings for all games and ratings for only successful games is significant at $p < 0.001$ indicating again users' strong sensitiveness to task success.

Figure 6.9 illustrates the user satisfaction results for all games as a set of box plots. We can see the user satisfaction results for the isolated policy on the left-hand side and the user satisfaction results for the joint policy on the right-hand

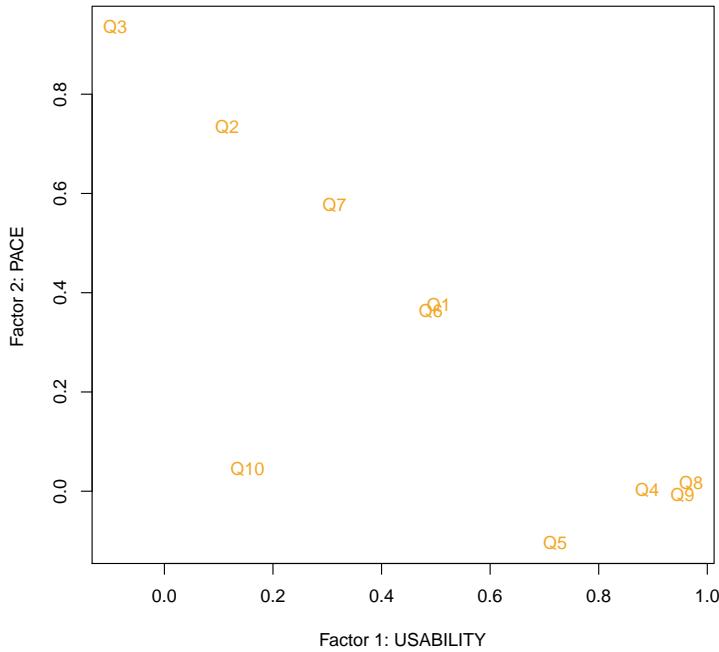
6. EVALUATION

side. The box plots can be interpreted as follows. The central mark represents the median of a range of data. The edges of each box are marked by the twenty-fifth and seventy-fifth percentile. The dotted lines that stretch outside the box represent the most extreme data points that are not considered outliers. Outliers are plotted individually as small crosses. Overall we can see two trends in the data. Those ratings of the isolated policy that stretch over a larger range of values than the joint policy point towards more negative ratings (Q1, Q4 and Q10). However, those ratings of the joint policy that stretch over a larger range of values than the isolated policy point towards more positive ratings (Q5, Q8 and Q9). Overall, users perceive the joint policy more positively than the isolated policy.

Despite an overall trend that users seem to prefer the joint over the isolated policy, we were not able to report any significant differences. Related work on the evaluation of spoken dialogue systems suggests a factor analysis [Dzikovska et al., 2001; Möller et al., 2007; Wolters et al., 2009]. An explanatory factor analysis explains the variability found in a set of observed, correlated variables in terms of a set of unknown latent variables, or factors. These factors are often fewer than the initial set of variables and reveal those underlying subjective categories that users were concerned about in their ratings. The advantage of a factor analysis is often that it reveals those subjective experiences with a system that matter to users, rather than reflecting the system designer’s expectations—as is often the case with pre-defined questionnaires. Please see Hone and Graham [2000] for details on a factor analysis applied to spoken language processing. A factor analysis applied to our subjective metrics of the GIVE evaluation showed the following. An illustration is provided in Figure 6.10.

Two factors were identified as accounting for 65% of the variability found in user ratings. For Factor 1, which we can call *Usability*, subjective metrics (Q4) ‘What to Do’, (Q8) ‘Enjoy Game’ and (Q9) ‘Recommend to Friend’ had high factor loadings of > 0.80 . Factor loadings indicate the correlations between questionnaire items. For Factor 2, which we can call *Pace*, only subjective metric (Q3) ‘Interaction Pace’ had a high factor loading of > 0.80 . The difference between the joint and the isolated policy for factor *Pace* was not significant at 0.9. While the difference for factor *Usability* was not significant either, at

6. Experimental Setting for Navigation in Virtual Environments



$p < 0.07$, at least, we can observe a statistical trend for this factor. All in all, it seems that statistical significance may have been achieved here if more data was available.

Effects of Game Worlds In addition to a joint or an isolated optimisation we can again observe strong effects of the individual game worlds on user satisfaction ratings. They are summarised in Table 6.6 for all games (in the upper part of the table) and for the subset of successful games (in the lower part of the table). The first three numeric columns summarise the average ratings per world (averaged over joint and isolated games) and the last three columns show the p -values for the comparison. Users assigned the highest ratings to games in World2 and the lowest ratings to games in World3. We can make the following observations.

- Users prefer games in World2 over World1 (nearly significantly, but not

6. EVALUATION

quite, at $p < 0.07$). This can be interpreted as evidence that the more robust a learnt policy is in terms of referring expression generation (the focus of World2), the better it is perceived by users.

- Users prefer games in World1 over games in World3 ($p < 0.0008$). In particular, they have a preference for the ‘Interaction Pace’ (Q3) in World1 over World3 ($p < 0.05$). This means that whenever a system proceeds too quickly and thus perhaps exceeds users’ cognitive load by changing instructions too quickly, they can get frustrated or confused.
- Users prefer games in World2 over games in World3 ($p < 0.0001$). This again is in accordance with the objective metrics that made it seem likely that the agent has learnt a more robust policy for referring expression generation (the focus of World2) than for navigation (the focus of World3). This tendency is confirmed by the subjective metrics.

While again the effects of individual game worlds are clearly visible, they cannot exclusively account for all the variation observed in user ratings. The joint versus isolated optimisation also exerted a visible influence especially with respect to the task success that the joint and the isolated policy achieved on average. We could observe clear differences between the user satisfaction ratings of successful games versus all games ($p < 0.001$).

6.2.3.3 Comparison with Systems from the GIVE Challenge

To allow for a comparison of our hierarchical RL framework with other state-of-the-art approaches to situated NLG, Table 6.7 contrasts our results with objective and subjective metrics collected for several systems in the GIVE-2 and GIVE-2.5 challenges. The former was run in 2010 and collected games from 1825 participants. The latter was run in 2011 and collected 536 games. The official results were discussed in Koller et al. [2010] and Striegnitz et al. [2011], respectively. GIVE-2.5 was run with the same evaluation worlds as our evaluation. The worlds in GIVE-2 were comparable in that all three worlds posed different challenges for the systems. World 1 was designed to be most similar to the training

Table 6.6:

User satisfaction results for the subjective metrics **per world** for all games (in the upper part of the table) and for just the subset of successful games (in the lower part of the table). Numbers refer to averages and are given together with their standard deviations. Note that for (Q10) ‘Language Robotic’, the results were inversed, so that higher number still indicate better ratings.

Subjective Metric	World1	World2	World3	p-val.	p-val.	p-val.
	W1	W2	W3	W1-W2	W1-W3	W2-W3
All Games						
(Q1) Easy to Understand	3.46 ± 1.12	3.63 ± 0.89	2.94 ± 1.02	.6	.2	.03
(Q2) Task Easy	3.0 ± 1.06	3.57 ± 1.07	2.47 ± 0.87	.1	.1	.001
(Q3) Interaction Pace	3.13 ± 1.5	3.42 ± 1.21	2.21 ± 1.18	.5	.05	.003
(Q4) What to Do	3.2 ± 1.14	3.42 ± 0.96	3.0 ± 1.21	.5	.6	.2
(Q5) Expected Behaviour	3.53 ± 1.24	3.78 ± 1.03	3.47 ± 1.07	.5	.9	.4
(Q6) Future Use	2.6 ± 1.05	2.94 ± 0.78	2.21 ± 0.61	.3	.2	.001
(Q7) Appropriate Help	3.2 ± 1.2	3.57 ± 1.07	2.57 ± 0.96	.3	.1	.005
(Q8) Enjoy Game	2.93 ± 1.03	3.05 ± 0.97	2.42 ± 1.18	.7	.2	.07
(Q9) Recommend to Friend	2.93 ± 1.03	3.05 ± 1.17	2.47 ± 1.17	.8	.2	.1
(Q10) Language Robotic	3.4 ± 0.73	3.05 ± 1.12	3.47 ± 1.21	.3	.8	.3
Sum	31.38	30.06	27.23			
Successful Games						
(Q1) Easy to Understand	3.61 ± 1.12	3.75 ± 0.85	3.11 ± 1.05	.7	.3	.1
(Q2) Task Easy	3.23 ± 0.92	3.68 ± 0.87	2.7 ± 1.0	.2	.2	.01
(Q3) Interaction Pace	3.46 ± 1.33	3.5 ± 1.15	3.0 ± 0.86	.9	.4	.3
(Q4) What to Do	3.38 ± 1.04	3.56 ± 0.96	3.11 ± 1.26	.6	.6	.3
(Q5) Expected Behaviour	3.69 ± 1.25	3.75 ± 1.12	3.44 ± 1.01	.9	.6	.5
(Q6) Future Use	2.84 ± 0.89	3.06 ± 0.77	2.22 ± 0.44	.5	.06	.006
(Q7) Appropriate Help	3.53 ± 0.87	3.75 ± 0.93	3.0 ± 1.0	.5	.2	.07
(Q8) Enjoy Game	3.15 ± 0.89	3.18 ± 0.98	2.67 ± 1.32	.9	.3	.3
(Q9) Recommend to Friend	3.15 ± 0.89	3.25 ± 1.12	2.78 ± 1.3	.8	.4	.4
(Q10) Language Robotic	3.46 ± 0.78	3.12 ± 1.14	3.88 ± 1.26	.4	.3	.1
Sum	33.5	34.6	29.91			

6. EVALUATION

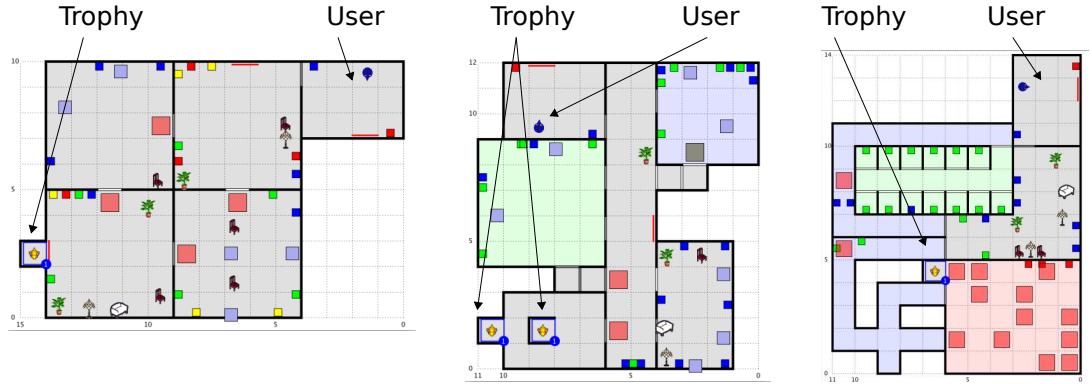


Figure 6.11: Evaluation worlds of the GIVE-2 challenge that was run in 2010 [Koller et al., 2010]. Again, World 1 was designed to be most similar to the training worlds by representing a medium degree of difficulty in navigation and referring expression generation. World 2 contained more complex referring expression generation scenarios and World 3 was challenging in terms of navigation.

worlds, while World 2 focused on referring expressions and World 3 on navigation. Figure 6.11 shows a map of the three evaluation worlds used in GIVE-2. All evaluations were thus carried out in comparable, if not identical, virtual worlds. All subjective scores in the table were rescaled from the -100 to $+100$ scale used in GIVE to our 1 to 5 scale.

Seven systems will be included in the comparison, the two best systems of GIVE-2 (**N**A and **S**) and the five best systems from GIVE-2.5 (**P1**, **P2**, **C**, **CL** and **L**). Since the overall results of GIVE-2.5 were better than of GIVE-2, more systems from the latter challenge are included in order to make a more challenging comparison.¹ Concretely, let us compare the following systems:

- one system from INRIA Grand-Est in Nancy [Denis, 2010] (**N**A),
- one system from Saarland University [Braunias et al., 2010] (**S**),
- two systems from the University of Potsdam [Garoufi and Koller, 2011b] (**P1/P2**),
- one system from LORIA / CNRS [Denis, 2011] (**L**),

¹No systems from GIVE-1 are compared because the setup of the first study was different in that users were not able to move through the environment freely, but had to move in discrete steps.

6. Experimental Setting for Navigation in Virtual Environments

- one system from the Universidad Nacional de Córdoba [Racca et al., 2011] (**C**), and
- one joint system from the Universidad Nacional de Córdoba and LORIA/CNRS [Benotti and Denis, 2011a] (**CL**).

While systems **NA**, **S**, **L** and **C** rely on rule-based solutions to GIVE, systems **P1** and **P2** use planning and system **CL** uses a corpus-based selection approach.

Since we are comparing data from separate evaluations, the results in Table 6.7 serve more as an indication rather than a direct comparison. Statistical significance is therefore not reported. There is unfortunately not always a perfect match between subjective metrics, but we wanted to include them nevertheless for a more comprehensive point of comparison. In particular, not all questions that we asked participants were the same that GIVE participants were asked. For category Q3, while we asked subjects *Was the speed of the interaction okay?*, GIVE asked participants to rate the statement *The system's instructions were visible long enough for me to read them*. For category Q4, we asked *Did you know at each moment what to do?* while the GIVE questionnaire contained *I was confused about which direction to go in*. Finally, while we asked *Did the system give you appropriate help when you needed it?* for category Q7, GIVE used *The system immediately offered help when I was in trouble*. All objective and other subjective categories have a direct correspondence. Unfortunately, the number of questionnaire items differed in GIVE-2 and GIVE-2.5, so that some fields in the table cannot be compared. We can make a number of observations from the data comparison:

- In terms of task success, we can see that our joint policy outperforms all other systems by at least 10%. This result also holds for other GIVE systems which were published separately from the challenge, such as Garoufi and Koller [2010] who achieve 69% and Benotti and Denis [2011b] who achieve 70%. This result reflects our reward function which placed a substantial weight on task success, rather than other metrics such as instruction or interaction length.
- The other objective metrics seem to suggest that both of our systems generate significantly more instructions and are more verbose than the other

Metric	Sys								
Objective Metrics	J	I	NA	S	P1	P2	L	C	CL
Binary Task Success	0.80	0.61	0.47	0.40	0.66	0.65	0.68	0.70	0.58
Duration (sec.)	632	781	344	467	407	415	341	538	539
Instructions (no.)	312.3	329	224	244	214	235	211	254	183
Words (total)	3075.6	4024.3	1408	1343	1122	1139	962	1328	1269
Subjective Metrics	J	I	NA	S	P1	P2	L	C	CL
(Q1) Easy to Understand	3.4	3.26	4.05	3.95	/	/	/	/	/
(Q3) Interaction Pace	2.9	2.86	2.65	2.5	3.42	3.45	3.77	3.55	2.82
(Q4) What to Do	3.43	2.91	3.02	2.57	3.15	2.9	3.2	3.8	3.17
(Q7) Appropriate Help	3.3	2.87	3.3	2.3	3.7	3.37	3.45	3.8	2.9
(Q8) Enjoy Game	2.95	2.56	2.3	2.3	/	/	/	/	2.6*
(Q9) Recomm. to Friend	3.0	2.56	1.75	1.9	/	/	/	/	1.8*
(Q10) Naturalness	3.36	3.21	2.4	2.6	/	/	/	/	3.2*

Table 6.7: Objective and subjective metrics for our systems (J=Joint and I=Isolated) compared with the best systems of the GIVE-2 challenge (NA and S) and the GIVE-2.5 challenge (P1, P2, L, C, CL). * indicates measures taken from [Benotti and Denis \[2011b\]](#) rather than the GIVE challenge.

6. Experimental Setting for Navigation in Virtual Environments

GIVE systems, which led to longer interaction times. This reflects the generation strategy learnt by our system, which was able to combine high-level and low-level instructions and aggregate several instructions into one. This produced many instructions such as *Go left and then towards the blue button*. In contrast, many GIVE systems relied predominantly on shorter instructions such as *turn left* or *press blue*.

- In terms of the subjective metrics, we can see that our system is slightly outperformed in *Easy to Understand* and *Interaction Pace*. The latter was already indicated in our own evaluation, where participants wished that instructions were displayed slightly longer and the system would reduce its overall interaction speed. On the other hand, our system performs substantially better in *What to Do* than most competitors and was ranked in the middle for *Appropriate Help*.
- We can further see that participants considered our system’s instructions noticeably more natural than its competitors’, enjoyed playing more and would recommend the game to a friend more often. In terms of naturalness, this is again reflected in our reward function, where we placed an explicit weight on human-like surface forms. To an extent, the other metrics confirm our earlier results in that participants enjoy playing when they win the trophy and they do not enjoy playing when they lose. Participants may therefore have enjoyed playing with our system most because it achieved the highest task success score overall.
- Finally, we can see that while the isolated policy is outperformed in many categories, it is still able to compete with some systems, such as in the categories *Interaction Pace*, *Enjoy Game*, *Recommend to Friend*, *Naturalness* and *Binary Task Success*. This indicates that even a policy optimised in isolation represents a competitive baseline.

The highest overall scores in this comparison were achieved by two rule-based systems, **C** [Racca et al., 2011] and **L** [Denis, 2011]. This suggests that a carefully designed ad hoc solution to a problem can still outperform many data-driven approaches to NLG nowadays. Systems **P1/P2** [Garoufi and Koller, 2011b] and **CL** [Benotti and Denis, 2011a] represent more state-of-the-art approaches. System

6. EVALUATION

P1 was using a combination of planning and supervised learning to NLG that aimed to maximise the understandability of referring expressions (**P2** acted as a planning-only baseline). This system received good scores for *Interaction Pace* and *Appropriate Help*, possibly because its planning steps guided users in small steps avoiding confusions and maximising understandability. System **CL** used a corpus-based selection approach, choosing instructions from a pre-collected corpus of human utterances in the same domain. This system was rated well for *Naturalness*. The reason is probably that it relied on instructions that humans produced for the very same situation the system was facing. On the other hand, this method does not take context into account which can lead to inconsistencies and low scores in other subjective categories. In summary, the comparison with these systems shows that our hierarchical RL approach is able to achieve comparable performance to state-of-the-art systems: while our joint policy is outperformed in some subjective categories, it achieves higher task success and more enjoyable and natural interactions than the other systems. This corresponds to the optimisation metrics that our reward function was designed for.

6.2.3.4 Discussion of Results

We had formulated the initial hypothesis that a jointly optimised NLG policy can lead to more successful, efficient and natural interactions that are better perceived by human users than their isolated counterparts. A human evaluation study has provided support for this hypothesis from several perspectives. Task success metrics showed that a joint optimisation leads to significantly more successful interactions, with significantly fewer problems, than an optimisation in isolation. These results confirm the simulation-based results reported in Chapters 3 and 4, where the isolated policy produced largely successful, but non-optimal, interactions. This sub-optimality was at least partially caused by the content selection and utterance planning decisions that the system had learnt in isolation of each other. Both modules face a trade-off between generating efficient interactions and easing the user’s cognitive load (at the cost of efficiency). It is therefore important that both modules work together. For example, if content selection decides to include fine-grained details into an utterance, utterance planning should choose to

6. Experimental Setting for Navigation in Virtual Environments

present the verbose semantic content in a way that eases the user’s cognitive load (for example, by choosing a one-by-one mode of presentation). Similarly, when utterance planning chooses an efficient way of presentation (such as presenting all the information jointly), then content selection should choose a concise semantic form. Whenever the trade-offs of efficiency and cognitive load are not balanced, interactions will become problematic which can be visible in the form of a lower binary or graded task success, increased number of system or user turns, or an increased number of false user actions due to confusion.

In addition to the task success results, evidence in favour of a joint optimisation is provided by the other objective metrics. The joint optimisation led to fewer system turns, system words and system words per turn. Interactions also took less time to complete. The isolated system apparently generates verbose language at the cost of the user’s cognitive load. This is a possible interpretation of the higher number of user turns in the isolated than in the joint setting. An exception is presented only by the number of wrong user button presses. While this number is higher in the joint setting, though, it also has a high standard deviation. One could thus say that while few users pressed a very high number of wrong buttons in the joint setting, the vast majority of users pressed none or only very few wrong buttons. This seems particularly likely given the higher task success results of the joint policy.

The user satisfaction ratings confirm the positive tendency of the objective metrics in favour of a joint optimisation. Users rate the jointly optimised system better than the system that was optimised in isolation. For this, we also discovered a statistical trend in an explanatory factor analysis. Among others, they express that the jointly optimised system was more easy to understand, gave more appropriate help in tricky situations and proceeded with a more appropriate interaction pace. These effects can likely again be traced back to the differences between content selection and utterance planning decisions learnt by each system. In addition, users rated the jointly optimised system, using graphical models to support surface realisation, as more natural and less robotic than its isolated counterpart that greedily exploits the most likely sequences of the language model. While this effect was not significant in the data, it is still remarkably consistent in the user ratings given that they were focused on solving a

6. EVALUATION

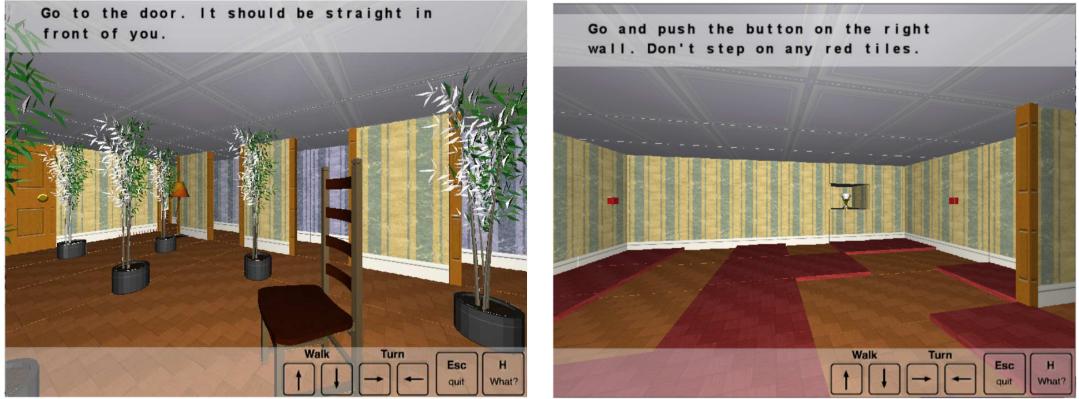


Figure 6.12: Two example situations (unseen from training) from evaluation world 3. On the left, the system generates an ambiguous reference to a door. On the right, it generates a referring expression combined with a warning to not step on any red tile.

navigation task rather than rating the naturalness of language. We can thus say that users were at least subtly aware of the difference.

Apart from a joint or isolated optimisation, the individual game worlds had a clear influence on the outcome of interactions and user satisfaction. People performed best in World2 and worst in World3 as we can see from the objective metrics. Unsurprisingly, people also liked World2 most and World3 least as they express in their subjective ratings. Especially with regard to World3, Figure 6.12 shows two example scenes from the world that were difficult for both system and users. On the left-hand side, we see a scene where the system wishes to refer to a specific door ('straight in front of you'). Unfortunately, the reference is ambiguous due to the high number of distractors and the system's precisely literal interpretation of the direction. While for the system 'straight in front of you' means an exactly straight line across, it is unlikely that the user shares this interpretation in the scene. On the right-hand side of the figure, the user has nearly won the game (the safe is already unlocked and the trophy visible) but needs to carefully navigate through a field of red buttons each of which can make the user lose the game immediately when stepped on. In both situations, it is

6. Experimental Setting for Navigation in Real Environments

thus critical to provide the user with just the right amount of information.

While the game world definitely played a role in the user ratings of a system, the world cannot be made to account for the differences observed between the joint and the isolated settings exclusively. In fact, the main contributing factor to user satisfaction in the GIVE evaluation appears to have been task success. Users rate those systems well that guide them to win the trophy. The jointly optimised systems did this in 80% of all cases, but the isolated system in only 60% of all cases. The difference between the average user satisfaction ratings of all games and the average user satisfaction ratings of only the successful games were consistently in favour of the successful games.

Finally, a comparison of our results against the GIVE-2 and GIVE-2.5 challenges allowed us to contrast our model with a number of state-of-the-art situated NLG systems. While our system was outperformed in several metrics, it achieved higher task success than any other system and was rated as more natural by human users. These were exactly the metrics that our reward function placed the highest weight on.

6.3 Experimental Setting for Navigation in Real Environments

In this section, we will apply the NLG approach developed in this thesis to a new, but related, domain: the navigation in real environments. To this end, we will integrate our designed surface realisation and utterance planning components as they were into an existing dialogue system. The system’s purpose is to assist users in wayfinding in a university building that is generally considered complex to navigate, especially for new and infrequent visitors. For the evaluation of the integrated system, we formulate the hypothesis that our developed approach can be transferred to new (related) domains without great effort and can therefore be considered portable and generalisable. The dialogue system that we use for this part of the evaluation has been described in detail in [Cuayáhuitl \[2011\]](#); [Cuayáhuitl and Dethlefs \[2011a,b\]](#) for an in-advance route giving scenario. For this evaluation, the system has been additionally equipped for the in-situ route

6. EVALUATION

giving case.

6.3.1 Experimental Methodology

Since the approach we suggest in this thesis aims to be generalisable to different domains, we will test and evaluate this generalisability and domain transferability in this section. To this end, we will address a navigation task in a real environment that corresponds to the setting we described in Section 3.5.2.1, the setting that underlies our reward function. Table 6.8 shows all the objective metrics used for the present study. They focus on measuring the time that users took to complete an interaction task and on measuring the binary and graded task success of interactions. Some other metrics typically used in the PARADISE framework [Walker et al., 1997, 2000] for evaluating dialogue systems were excluded due to their lack of relevance to the evaluation scenario. Since users interact with the system exclusively using button presses, the number of user or system turns, for example, does not serve as a good indicator of the interaction efficiency. All information relevant to the objective metrics was noted down by an assistant that followed users during their wayfinding tasks. With respect to interaction time, we measured the time that users took from leaving their origin to their destination. The time was stopped when users had reached the destination (or gave up the navigation task). In addition, the assistant took note of the difficulties that users encountered during their interactions. They noted down a value of ‘no problems’ if users found their target location immediately, a value of ‘slight problems’ if users hesitated briefly but then found the target location and a value of ‘severe problems’ if users eventually found the target location but only after getting lost or hesitating several times. When the user did not find the target location, this was also noted down. The times taken and the values assigned by the assistant show a correspondence indicating that the assistant assigned reasonable scores. To measure the task success of different interactions, we use binary task success, on the one hand, and the 3-valued and 4-valued graded task success metrics introduced in Section 6.2.1, on the other hand.

Table 6.9 maps subjective metrics we wish to evaluate with the questions that were raised to users in order to obtain them. The metrics were obtained using

6. Experimental Setting for Navigation in Real Environments

Table 6.8: Objective metrics for the real navigation evaluation. We use category O1 to measure the time that an interaction took and measures O2, O3 and O4 to measure the success and smoothness of interactions.

Measure	Description
Dialogue Efficiency	
(O1) Session Duration	How long was the session in seconds?
Task Success	
(O2) Binary Task Success	Was the game won or lost (giving up counts as lost)?
(O3) 3-Graded Task Success	Was the game won without problems, won with small problems, or other?
(O4) 4-Graded Task Success	Was the game won without problems, won with small problems, won with severe problems or lost?

Table 6.9: Subjective metrics for the real navigation evaluation and the questions that were asked in order to obtain them.

Measure	Question
(Q1) Easy to Understand	Was the system easy to understand?
(Q2) System Understood	Did the system understand what you asked?
(Q3) Task Easy	Was it easy to find the location you wanted?
(Q4) Interaction Pace	Was the pace of the interaction appropriate?
(Q5) What to Say	Did you always know what you could write?
(Q6) Expected Behaviour	Did the system work the way you expected?
(Q7) Future Use	Would you use the system in the future?

a questionnaire that users were asked to fill following each complete interaction with the system, i.e. following one navigation task.

6.3.2 Experimental Setup

14 participants took part in the evaluation study, 93% (13 out of 14) were female and 7% (1 out of 14) were male. All participants took part either against payment or to fulfill partial course requirements. Participants interacted with the system using an Android mobile phone application that guided them to different destinations in a university building (which is generally considered complex to

6. EVALUATION

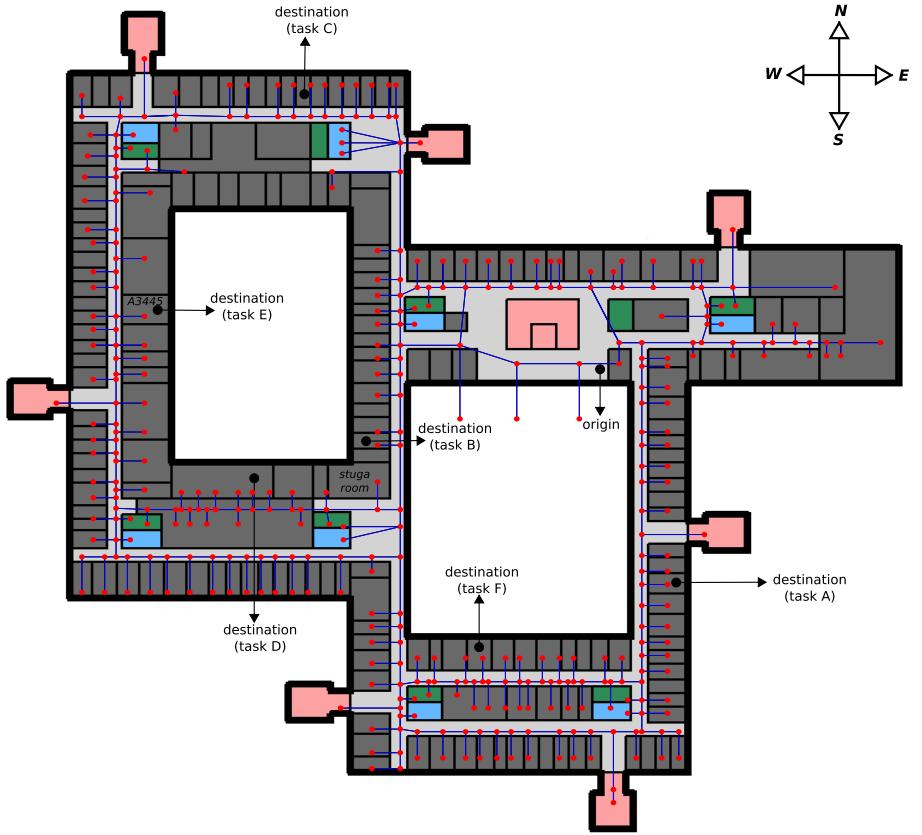


Figure 6.13: Map of the navigation environment for the in-situ evaluation in a real setting. The six destinations that participants had to navigate to are shown as destination (a)–(f) on the map. The order of tasks was randomised.

navigate) using text-based natural language.¹ After a short training in using the phone-based application, participants had to complete six navigation tasks. They asked the system for route instructions to their destination using natural language with an auto-fill mechanism for location names. The system then provided them with route instructions for the next route leg from their current location towards the destination. The first instruction was always given from the origin (cf. the

¹We decided to use a text-based version of the system, rather than a speech-based one, since the latter could introduce a large number of speech recognition errors that can be regarded as noise to the evaluation of an NLG system that was not designed to deal with speech recognition results. The text-based version allowed us to fully concentrate on the relevant NLG behaviours.

6. Experimental Setting for Navigation in Real Environments

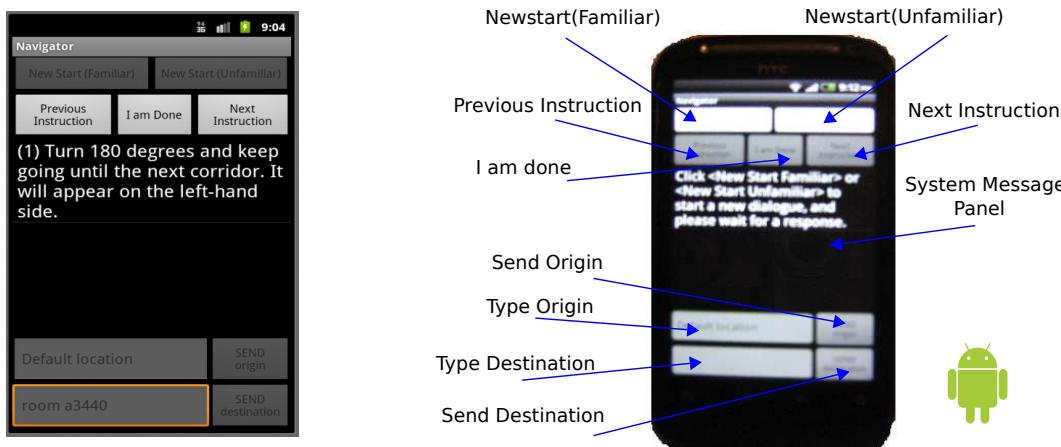


Figure 6.14: The Android application that participants used to navigate to a set of six different destinations. On the left: the simulated phone used for development. On the right: the actual phone used during the evaluation with arrows pointing to buttons or text fields together with the description of their purpose. All experiments were run on the HTC Desire S mobile phone using Android 2.3.

map in Figure 6.13). Participants would navigate the instruction and press the *next* button in order to see the next instruction or the *previous* button in order to see the previous instruction. The system itself did not use GPS to track users, but relied on that users would ask for instructions (by pressing the *next* button) at each relevant decision point. All participants took part in six navigation tasks drawn randomly from a uniform distribution. A map of the evaluation environment with all six destinations is shown in Figure 6.13.

Figure 6.14 shows a photo of the mobile phone that participants used during the evaluation. It was an HTC Desire S that ran Android 2.3 (Gingerbread). Participants had seven buttons available and two text fields. The buttons and their descriptions are shown in the picture. Participants that were unfamiliar with the navigation environment used the button ‘Newstart(Unfamiliar)’ to start a new interaction. Participants that were already familiar with the environment used the ‘Newstart(Familiar)’ button.¹ Participants of this evaluation were largely unfamiliar with the building. To specify a new destination, participants typed the

¹In this evaluation, both buttons had the same functionality. In the future however, the two buttons could evoke different system behaviours based on the user’s prior knowledge.

6. EVALUATION

name of the destination into the ‘Type Destination’ text field and then clicked the ‘Send Destination’ button. Origins could be specified in the same manner (using the ‘Type Origin’ text field and the ‘Send Origin’ button), but this feature was not used in the evaluation. Once a route had been obtained, participants could press the ‘Next Instruction’ button to see the next instruction and the ‘Previous Instruction’ button to see a previous instruction. When they had found their destination, they could end the interaction by pressing the ‘I am done’ button.

6.3.3 Experimental Results

Following the evaluation of a mobile application that generates in-situ route instructions to human users, we analysed the collected data. The results of the objective and subjective metrics are discussed in the following.

6.3.3.1 Objective Metrics

The results for the objective metrics are summarised in the upper half of Table 6.10. There were no unsuccessful navigation tasks in this scenario, users always eventually found their target location. More specifically, 60% of all interactions were completed without problems, 30% were completed with slight problems and 10% with severe problems. These results allow us to draw some conclusions with respect to an in-advance versus in-situ instruction giving scenario. The baseline system presented in Cuayáhuitl and Dethlefs [2011a] has the same functionality as our system but gives instructions in-advance rather than in-situ. This means that instructions for the entire route to follow are presented to the user before navigation starts. The user then has to either memorise the route or take notes. In the latter case, the time that users took notes was not included in the overall interaction time with the system during wayfinding. Using the in-advance setting, users achieved a binary task success of 93%. This is significantly lower than in the in-situ scenario ($p < 0.02$). They completed 75% of all interactions without problems (which is more than in the in-situ scenario at $p < 0.04$), 11% with slight problems ($p < 0.001$) and 14% with severe problems (not significant at $p < 0.29$). Figure 6.15 presents bar plots to illustrate the comparison. We can see that the in-situ system receives the best binary task success results. In terms of 4-valued

6. Experimental Setting for Navigation in Real Environments

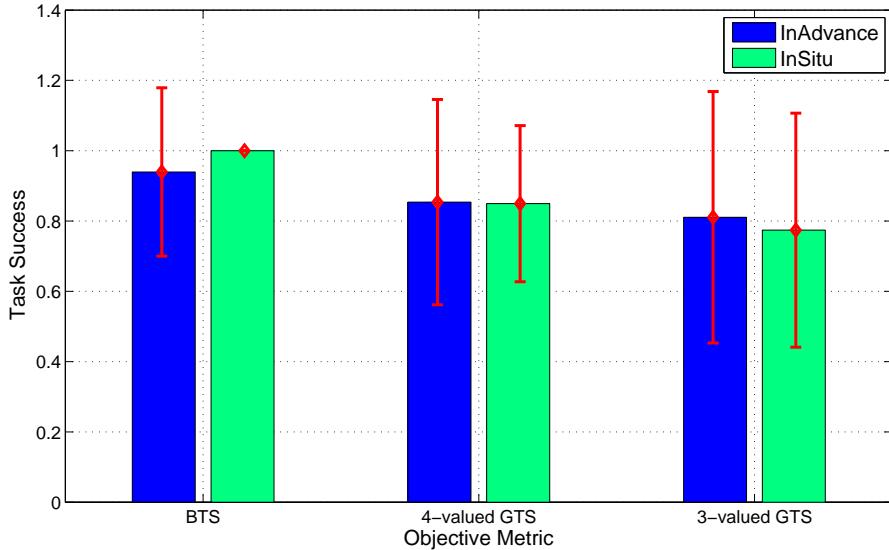


Figure 6.15: Bar plots illustrating the binary and graded task success for the in-situ wayfinding scenario evaluated in this thesis in comparison to the baseline in-advance wayfinding scenario reported in [Cuayáhuitl and Dethlefs \[2011a\]](#). The first two metrics, binary task success and 4-valued graded task success, seem to suggest that users are more successful in the in-situ scenario. The 3-valued graded task success metric in contrast points in the opposite direction suggesting that users encounter less problems in the in-advance scenario.

graded task success, both systems perform about equally, but the in-situ system has a smaller standard deviation. The 3-valued graded task success metric, on the other hand, seems to point into the opposite direction suggesting that in-advance interactions were slightly more successful. Since users in both evaluation scenarios, in-advance and in-situ, navigated to the same set of destinations, we can also use the interaction time as an indicator of possible problems that users encountered. Figure 6.16 provides bar plots for the average time that users took to complete a wayfinding task in each scenario. While in the in-situ scenario, users took on average 1 minute and 42 seconds, in the in-advance scenario they took on average 2 minutes and 23 seconds. This is a difference of 41 seconds per task. Both sets of results in combination, elapsed average time and task success, provide evidence that users are more successful in an in-situ navigation scenario

6. EVALUATION

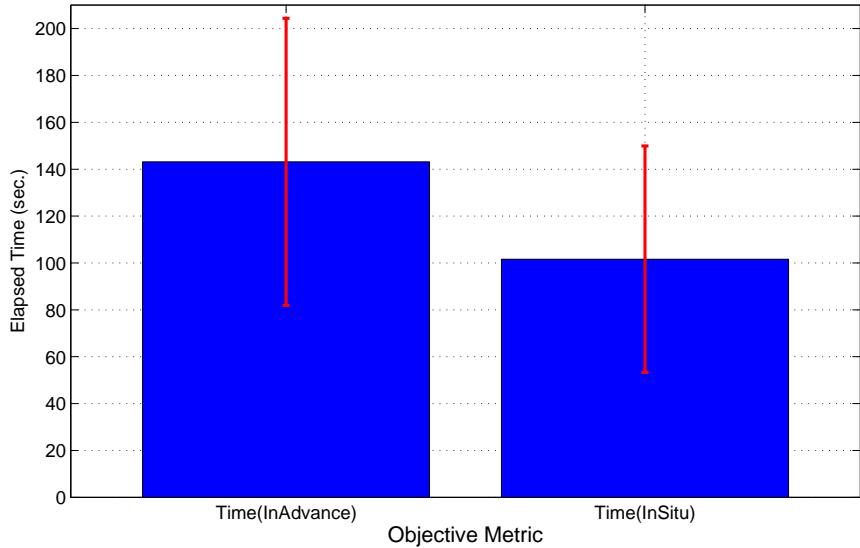


Figure 6.16: Bar plots comparing the average time taken by users to complete a wayfinding task in the in-situ scenario evaluated in this thesis (on the right) and in the baseline in-advance scenario (on the left) [Cuayáhuitl and Dethlefs, 2011a]. Interactions are visibly shorter and more efficient in the in-situ scenario. The difference corresponds to 41 seconds on average.

than in an in-advance navigation scenario.

6.3.3.2 Subjective Metrics

Following each interaction task with the mobile application, users were asked to fill a questionnaire indicating their user satisfaction with the interaction. The lower half of Table 6.10 summarises the subjective metrics collected in this way per navigation task. On the whole, we can see that the system is rated very well. The best rated metric is ‘Interaction Pace’(Q4) which receives an average of 4.81 (± 0.5). The worst rated category is ‘Future Use’ (Q7) with an average rating of 4.25 (± 0.9). The difference between these two shows a significant trend ($p < 0.06$). The users indicated that they found the system easy to use and easy to understand. They felt that the system was giving them appropriate guidance and reacted promptly and adequately to their queries.

6. Experimental Setting for Navigation in Real Environments

Table 6.10:

Results for the objective and subjective metrics of the in-advance and in-situ interactions. Numbers refer to averages. Subjective metrics were again rated on a 1-5 Likert scale, where 5 represents the best and 1 the worst score. The results for the in-advance instructions were taken from Cuayáhuitl and Dethlefs [2011a].

Objective Metric	In-Advance	In-Situ
(O1) Session Duration	2:23	1:42
(O2) Binary TS	93.4	100.0
(O3) 3-Graded TS	81.0	0.77
(O4) 4-Graded TS	86.9	0.84 2
Subjective Metric		
(Q1) Easy to Understand	4.52	4.54
(Q2) System Understood	4.84	4.75
(Q3) Task Easy	4.24	4.43
(Q4) Interaction Pace	4.56	4.81
(Q5) What to Say	4.74	4.49
(Q6) Expected Behaviour	4.40	4.51
(Q7) Future Use	3.82	4.25
Sum	31.12	31.78

With respect to the comparison between an in-situ and an in-advance navigation scenario, the user satisfaction results are in accordance with the objective metrics reported above. The difference in user satisfaction between the in-advance and in-situ scenario is significant at $p < 0.05$. The differences between both systems are illustrated as box plots in Figure 6.17. We can see that differences exist especially for the metrics ‘Interaction Pace’ (Q4), ‘What to Say’ (Q5) and ‘Future Use’ (Q7). Users perceived the interaction pace in the in-situ scenario as better than in the in-advance scenario. In contrast, they indicate that they knew less often what to say to the system in the in-situ setting than in the in-advance setting. A possible reason for this result is that the interaction between user and system in the in-situ setting was reduced to button clicks. Nonetheless, users rate their future use of the in-situ system more optimistically than their future use of the in-advance system.

6. EVALUATION

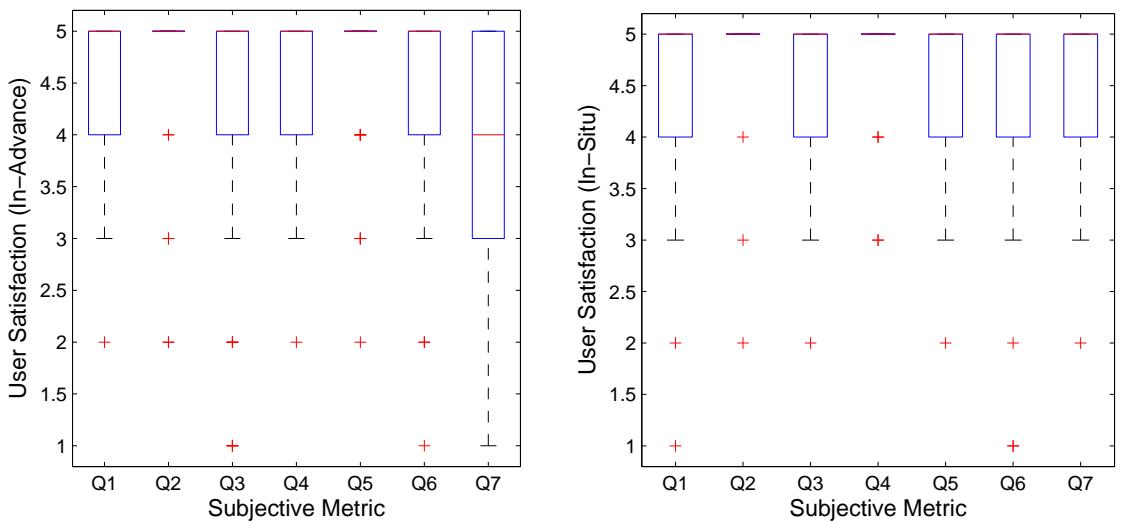


Figure 6.17: Box plots illustrating the user satisfaction ratings for the baseline in-advance scenario [Cuayáhuitl and Dethlefs, 2011a] (on the left) and the in-situ scenario evaluated in this thesis (on the right). Users preferred the interaction pace (Q4) in the in-situ scenario and expect to use the system more often in the future (Q7) than the in-advance system. On the other hand, they indicate that they knew less often what to say (Q5) to the system in the in-situ setting than in the in-advance setting. This may be due to the fact that interaction in the in-situ scenario was controlled predominantly by button presses.

6.3.3.3 Discussion of Results

In this section our aim has been to test the transferability of the NLG methods developed in this thesis to new but related domains. To this end, we integrated the utterance planning and surface realisation components with an existing route planning and dialogue management system for indoor navigation. The evaluation with human users has shown that the combination can achieve high binary and graded task success scores as well as high user satisfaction scores. Especially the subjective metrics ‘Easy to Understand’ and ‘Interaction Pace’ received high scores of the human judges. While no claim is made with respect to the content selection of the system, which was dealt with by the dialogue management and route planning modules, the transfer of the utterance planning and surface realisation components can be regarded as successful in this context. Moreover,

6. Conclusion

we could observe that the transfer of the system to the in-situ scenario achieves substantially better task success and user satisfaction scores than a previous evaluation in an in-advance setting [Cuayáhuitl and Dethlefs, 2011a]. Figure 6.18 shows a sample interaction from the origin to destination F (cf. Figure 6.13 for a map). The interaction starts with the system giving the first instruction that is displayed under the first picture. The user follows the instruction and then presses the ‘Next Instruction’ button as they reach the first decision point (shown on the second picture). The system then generates the next instruction from the current decision point to the next. Once the user has followed the instruction and reached the next decision point, they can press the ‘Next Instruction’ button again. The system then generates the third pair of instructions (shown under the third picture) and the user executes them. Finally, as the user presses the ‘Next Instruction’ button again, the system generates the last instruction (shown under the fourth picture). As soon as a user has found their target destination, they can press the ‘I am done’ button to end the interaction.

6.4 Conclusion

This chapter has provided a comprehensive human study evaluating the methods designed in this thesis. We have compared our jointly optimised learning framework against a learning framework that was optimised in isolation. In summary, we have seen that the joint policy leads to more successful and more efficient interactions, encounters less problems and is perceived as better and more natural by human users. These results confirm the findings we have made in previous chapters that were based on simulation, preliminary user ratings or automatic metrics: humans prefer the jointly optimised policies over their isolated counterpart and express this in their system ratings.

We can further use this evaluation study as evidence for the general ability of reinforcement learning agents to adapt to new domains and behave reasonably in environments that they have not been trained in. Both the joint and the isolated policy were exposed to new worlds and new users in the evaluation, but displayed largely coherent behaviour all along. A purely decision tree-based baseline [Dethlefs, 2011] (such as we developed in Chapter 4) performed substantially

6. EVALUATION



User: [Next Instruction]

- (1) Turn around and go until the glass door in front of you.



User: [Next Instruction]

- (2) Turn left and go until the next corridor on your right.



User: [Next Instruction]

- (3) Bear right and go straight.



User: [Next Instruction]

- (4) You will see room A3850 on your right-hand side.

User: [I am Done]

Figure 6.18: An example interaction from the origin to destination E (see map in Figure 6.13) including four decision points. The photo in each case shows the decision point that the user faces at the moment of the next instruction. User button presses are shown in cursive fonts. The photos are a courtesy of Cui Jian.

6. Conclusion

worse in the GIVE-2.5 challenge in 2011 [Striegnitz et al., 2011]. In this system, the decision trees had also been trained on worlds different from the ones they were evaluated in, but then unfortunately failed to adapt to new circumstances and situations leading to bad task success and user satisfaction scores. Several systems that performed better in the challenge were based on planning [Garoufi and Koller, 2011a,b], corpus-based selection [Benotti and Denis, 2011b] or various sets of rules. None of the more successful systems was based purely on machine learning. The evaluation study presented in this chapter can thus also be seen as a demonstration of that reinforcement learning-based systems (with graphical models) can learn to adapt to their environments and achieve similar performance to systems that heavily draw on human design decisions.

In addition to testing the generalisability of reinforcement learning policies to new settings, such as a new game world, we also evaluated its transferability to new, but related, domains. We applied the utterance planning and surface realisation policies to a real navigation scenario in a university building. Results showed that the NLG component was able to contribute to high task success and user satisfaction scores. Especially the understandability of the system and the interaction pace were appreciated by human users. A comparison of the designed in-situ wayfinding system with a system that gave instructions in-advance (but had otherwise identical functionality) showed that users were more successful in the in-situ scenario and expressed an overall preference for the in-situ system in their user satisfaction ratings.

6. EVALUATION

Chapter 7

Conclusions and Future Research

7.1 Contributions and Findings

Natural Language Generation systems for interactive contexts are faced with numerous trade-offs in generating an utterance that is optimally adaptive to the current spatial situation, prior knowledge and degree of confusion of the user, the linguistic context and the interaction history. These trade-offs include the level of detail chosen in an utterance (how much information to give to a user?, how much prior knowledge to assume?), the presentation mode chosen for the utterance (impose a temporal structure on the utterance?, present it in a piece-meal fashion or all together?, what information structure to use?), and the surface realisation chosen for communication to the user (what lexical items to choose? whether to align with the choices of the previous utterance or vary them? if the user is confused, whether to repeat the previous utterance or paraphrase it?). The ultimate goal of each utterance is to communicate information effectively and efficiently with a minimal degree of cognitive load imposed on the user and a maximal degree of naturalness and adaptation to the user and context. In addressing these challenges, this thesis aims to make a contribution to the field of computational linguistics, in particular reinforcement learning, for robust, flexible and adaptive Natural Language Generation for interactive systems. The main contributions of this thesis are summarised in the following paragraphs together with their main findings.

(1) The application of Hierarchical Reinforcement Learning to NLG

In order to achieve optimal and flexible adaptation to unseen users and situations, we have suggested to organise complex NLG tasks as hierarchies of sub-tasks and optimise them using hierarchical reinforcement learning. This idea was demonstrated with a hierarchical learning agent for the GIVE task, a situated interaction task in a virtual 3D environment. A flat learning agent for this prob-

7. CONCLUSION

lem had roughly 10^{57} state-actions which made the hierarchical decomposition beneficial. For each subtask in the hierarchy, we formulated a set of states to represent the agent’s knowledge about the world, in this case about the spatial setting, the user and the interaction history. A set of actions was defined to correspond to the agent’s behavioural potential, i.e. everything the agent can say or do. A probabilistic transition function specified in which way every action of the agent could change the world, and a reward function was induced from human data to reward the learning agent for *the shortest possible interactions at maximal task success*. A simulation-based evaluation of this framework was encouraging and showed that the agent learnt to adapt to unseen circumstances and users (given in the simulated environment) over time generating successful and smooth interactions.

(2) A joint optimisation of distinct NLG subtasks Decisions in many domains of the world are interrelated and typically considered in relation to each other. In NLG, the tasks of content selection, utterance planning and surface realisation have traditionally been treated as a sequence of modules making independent decisions. This thesis has argued that such an isolated treatment is inappropriate given the numerous interdependencies that exist between all three subtasks. As an alternative, a joint learning framework has been suggested in which decisions of all levels are considered and optimised jointly (by sharing information among them). The result is a behaviour policy that takes more trade-offs into account for generating interaction contributions than a policy that was optimised in isolation. These trade-offs include the spatial environment, the user’s information need and cognitive load and the likelihood of surface realisation variants with respect to a human corpus. A simulation-based optimisation showed that a jointly optimised policy achieves significantly higher *user satisfaction* than a policy optimised in isolation by taking these trade-offs into account.

(3) A Hierarchical Information State for constrained learning under the inclusion of prior knowledge Often the designers of generation systems have prior knowledge of their target domain that they may wish to include in a system. Furthermore human preferences may exist for many domains that can

7. Contributions and Findings

motivate the behaviour of NLG systems even if they do not have an impact on task success or user satisfaction. Such prior knowledge and human preferences also exist for the GIVE domain as we identified based on decision tree learning. To include them into our NLG model, we presented the notion of a hierarchical information state, where one information state is defined for each learning agent in the hierarchy. A hierarchical information state allows the systematic specification of rules that constrain the agent’s learning process and simultaneously lead to faster learning due to the constrained search space. A simulation-based optimisation showed that the policy learnt with the constrained action set is equivalent (in terms of average rewards) to a fully-learnt policy but can be learnt faster and increase the scalability of the proposed approach.

(4) The application of graphical models to context-sensitive surface realisation Surface realisation decisions are often based on a human corpus of the target domain such as the GIVE corpus for generating instructions in virtual environments. Such corpora can be used to induce generation spaces—spaces of surface realisation variants for a specific semantic concept—to motivate the system’s output decisions based on human examples. However, surface realisation decisions in the joint framework should not be treated in isolation but in conjunction with content selection and utterance planning. We have suggested to use graphical models, Hidden Markov Models or Bayesian Networks, as representations of generation spaces and integrate them tightly into our hierarchical joint learning framework. We trained one generation space for each surface realisation agent and used them to provide feedback—in the form of rewards based on the forward probability (for HMMs), or probabilities derived from probabilistic inference (in Bayesian Networks)—to the agent during its learning process. A simulation-based optimisation indicated that the graphical models contribute to user satisfaction by supporting surface realisation that is more sensitive to decisions of content selection and utterance planning.

(5) Optimisation of an alignment score based on Gaussian sampling The psychological theory of *interactive alignment* states that humans align their linguistic representations during interactions based on priming. Since the mental

7. CONCLUSION

representations used during language production and comprehension are assumed to be the same, humans are primed by their interlocutor’s utterances as well as by their own. The GIVE corpus contains abundant examples of alignment. However, humans also vary their surface realisation decisions—apparently just for variation’s sake. We suggested the notion of a *constituent alignment score* which indicates the proportion of variation and alignment and observed that it lies at roughly 0.5 for situated interaction. To include the *constituent alignment score* as one of the agent’s learning objectives, we sampled rewards from a Gaussian distribution where the highest rewards were obtained for a medium proportion of alignment and variation. An evaluation based on similarity with human authors indicated that the combination of Gaussian sampling and graphical models contributes to user satisfaction and natural surface realisation.

(6) A comparison of graphical models for surface realisation Since HMMs and Bayesian Networks have different theoretical and practical properties, two questions were raised for surface realisation: which of these models would perform best in a direct comparison? and how would both models perform in a comparison with a random or greedy baseline? To investigate these questions, we evaluated the user satisfaction and naturalness achieved by HMMs, Bayesian Networks, PCFGs (as a greedy baseline) and CFGs (as a random baseline) in a direct comparison within the joint learning framework and in isolation. We found that PCFGs and CFGs both led to only suboptimal system performance. In contrast, we saw that while Bayesian Networks performed best in isolation, HMMs achieved comparable user satisfaction and naturalness results within the joint learning framework. This indicates that the hierarchical learning agent discovers the non-optimal behaviour that is caused by the HMM’s lack of context-awareness (due to their independence assumptions) and learns to balance this drawback by learning a more comprehensive policy itself. HMMs are therefore an attractive and scalable model for complex and large applications within the proposed joint learning framework.

(7) A task-based human evaluation study Not all results that achieve significant effects in simulation or automatic metric-based evaluations are also per-

7. Contributions and Findings

ceived by humans. We therefore evaluated our optimisation framework based on joint hierarchical reinforcement learning including Bayesian Networks and Gaussian sampling in a task-based human evaluation study. Participants played with two systems—one optimised jointly and one optimised in isolation—in three different GIVE worlds which were different from the ones in which the learning agent was trained. The following findings were made: (a) the hierarchical learning agent with constrained actions had learnt a robust generation policy that adapted to new circumstances and users flexibly and led to (predominantly) smooth and successful interactions; (b) the jointly optimised system achieved substantially higher user satisfaction and task success results than the system optimised in isolation; (c) users consistently indicated a preference for the surface realisation choices based on Bayesian Networks and Gaussian sampling than for the purely frequency-based choices.

(8) A transfer study The approach developed in this thesis aims to be domain-independent and therefore generalisable and transferrable to different domains with limited effort. To demonstrate these properties, we transferred our agent to a new, but related, domain: the generation of route instructions in a real navigation scenario. The learnt policies for surface realisation and utterance planning were integrated into an existing in-situ dialogue system for wayfinding in a university building that is generally considered complex to navigate. Participants in an evaluation study carried the system on a mobile device that generates route instructions along the way at relevant decision points. The following findings were made: (a) the policies were transferable to the new scenario with limited effort; (b) they contributed to high user satisfaction, task success and smooth interactions; (c) in comparison to an in-advance route giving scenario, the in-situ setting achieved higher user satisfaction, task success and more efficient interactions, presumably due to a reduced cognitive load.

(9) A review of context-sensitive and machine learning-based NLG
The field of context-sensitive NLG has been approached from a number of different angles over the years, including early rule-based accounts, planning and constraint-satisfaction accounts, trainable accounts using supervised or unsuper-

7. CONCLUSION

vised learning and several others. Trainable approaches have gained increased significance over the last decade due to the shorter development times and lower costs involved in designing systems. Research in applying reinforcement learning for NLG has mainly focused on flat reinforcement learning settings, which can be a major limitation when developing real-world systems, since flat RL is notoriously affected by the *curse of dimensionality*, which limits its practical scalability. Related work in treating interrelated tasks jointly to achieve better performance has reported positive results, but a joint treatment of all aspects of content selection, utterance planning and surface realisation had not yet been attempted. Finally, we have reviewed research on using graphical models as generation spaces for surface realisation.

(10) Semantic annotations of the GIVE corpus Machine learning methods are powerful tools in the design of robust, flexible and adaptive systems for human-computer interaction. Unfortunately, they rely on large amounts of annotated data, which is often unavailable or expensive and time-consuming to obtain. One of the contributions of this thesis is a set of detailed semantic annotations of 61 English interactions (with over 300 turns) of the GIVE corpus that were done by two independent annotators. These annotations will be made available to the research community for analysis or automatic training of systems.

In a nutshell, the main contribution of this thesis can be summarised as *a joint optimisation framework using hierarchical reinforcement learning and graphical models for robust, flexible and adaptive NLG for situated interaction*. This thesis aims to make a contribution towards the development of adaptive interactive NLG systems. However, a lot remains for future research.

7.2 Future Directions

The field of machine learning for Natural Language Generation is still young with many exciting research directions to pursue in the future. Some of them—with a focus on reinforcement learning for NLG—are discussed briefly in the following

7. Future Directions

paragraphs.

Joint Optimisation The idea of jointly optimising a set of distinct but related subtasks is likely to improve the performance of systems generally—not just restricted to the domain of NLG. Just as it has been demonstrated to benefit systems using dialogue management and NLG [Lemon, 2011], dialogue management and route planning [Cuayáhuitl and Dethlefs, 2011a] or different components of NLG systems [Angeli et al., 2010; Dethlefs and Cuayáhuitl, 2011a,b], it will probably improve the performance of all systems in which important trade-offs exist among subtasks. Examples of related tasks that have been treated in isolation but may benefit from a joint treatment include: (a) language analysis and production where a joint treatment could help to align the discourse models of the system and the user at the semantic, grammatical and phonetic level; (b) the language processing components of an interactive system and domain-specific modules where a joint treatment could make database queries dependent of linguistic knowledge and vice versa; (c) the language processing components of an interactive system with multi-modal behaviours where a joint treatment could help to reinforce communicated contents with non-linguistic behaviour.

Online Learning and Adaptation Reinforcement learning agents typically learn a behaviour policy offline during a training phase in a simulated environment and then execute the learnt policy (in a static way) during testing. A frequent argument for not learning from interactions with real users is that usually agents need several thousands of interactions to learn a good policy for a domain, which makes learning from real interactions impractical. While this argument is valid, it seems to impose a significant restriction on the degree of adaptation that the learning agent can display towards individual users and interactions. In order to adapt to individual circumstances of the interaction, it would be most intuitive to learn from the individual interaction. This is even similar to the way that humans learn. To allow agents to learn from real interactions, however, more efficient training algorithms are needed that allow action values to be computed quickly and reliably so that they could immediately have an impact on the agent’s current behaviour. While first approaches towards online learning—learning dur-

7. CONCLUSION

ing and from the interaction—already exist [Cuayáhuitl and Dethlefs, 2011a,b; Gašić et al., 2011], more research in this direction is needed to make the learning behaviour of agents more efficient and suitable for online adaption.

Incremental Generation Incremental interactive systems are characterised by the property that they start to process, plan and generate an utterance while the user has not yet finished theirs [Skantze and Schlangen, 2009]. This characteristic is intended to make incremental systems more natural and intuitive to interact with because they employ a turn-taking mechanism that is similar to the way that humans take turns in interaction. Apart from turn-taking, though, several discourse phenomena go hand-in-hand with the incremental architecture such as backchannels, hesitations, interruptions or non-linguistic behaviours such as pointing [Buss and Schlangen, 2010]. All of these phenomena can help the analysis of discourse contributions (by supporting grounding) or their generation (by providing evidence on when to take the turn, when to generate a backchannel, etc.). Future work can investigate alternative system architectures for incremental systems [Buss and Schlangen, 2011; Schlangen and Skantze, 2011] and the cognitive and psycholinguistic effects they have on human interaction processing or on human-computer interaction. Incremental discourse processing can also be treated as an optimisation problem in the future to balance the trade-offs that arise with newly emerging and competing system actions. This direction is especially interesting for incremental NLG which has been hand-crafted so far [Skantze and Hjalmarsson, 2010].

Automatic Agent Induction Reinforcement learning agents are typically designed by a system developer who bases his or her design decisions on knowledge of the task, the domain or the end user of the system. Two drawbacks come with this procedure. The first is that system development can be slow because it requires labour-intensive human analyses of the target domain. The second, and perhaps more serious, drawback is that different design decision in system development can have different effects for the performance of a system. This begs several questions about the best system design, the evaluation of a chosen design and the comparability of systems with different designs. An interesting direction

7. Future Directions

for future research is the investigation of methods to induce system designs from human or domain data automatically. For reinforcement learning in particular we would like to automatically learn the distinguishing features of a domain that should be included in the agent’s state space and those that should be included in the agent’s action set. While frameworks exist to induce a reward function from data [Walker et al., 2000], more research is also needed in this area. In addition, we also would like to induce the agent’s hierarchical structures from data rather than specifying them purely based on human intuition.

Dynamic Learning Agents Using a *divide-and-conquer* approach to divide a large optimisation problem into a set of smaller ones as in hierarchical reinforcement learning has a number of advantages. It reduces computational complexity by avoiding the *curse of dimensionality* problem, i.e. the problem of exponential growth of the state space according to the number of variables. Moreover, by organising a large problem into a number of well-motivated subproblems, we may structure the task in a way that makes it more intuitive and simple to use for human users. Unfortunately, imposing a structure over a task usually also means that both system and user have to follow this structure throughout the interaction—in a sometimes too rigid way. For interactive systems, for example, the user is not allowed to change the current focus of the interaction if the new focus is not located within the current subtask. This is a general limitation of current approaches that use a module-based system architecture without flexible communication between modules. Future work should investigate more flexible system architectures that facilitate the transition between subtasks and states by allowing them to change dynamically. This is especially true when NLG systems are tightly coupled with other components (e.g. dialogue management, route planning, multimodal behaviours) and aim to perform learning during the course of the interaction.

Generation under Uncertainty Reinforcement learning agents for NLG currently always make the simplifying assumption that their knowledge about the user and the environment is complete. An advantage of this assumption is that there is no uncertainty with respect to the state that the agent is in and therefore

7. CONCLUSION

of the best action to take. Realistically speaking however, this assumption is problematic because most environments in which the agent acts are not fully observable. With respect to the GIVE domain, for example, we have performed action selection based on the user’s degree of confusion. While we made the assumption that we knew the user’s state of confusion at all times, it is unrealistic to assume this. All we can usually do in the real world is to make assumptions about the user’s most likely state of confusion (and therefore also about the most likely best action to take). While research on Partially-Observable environments has been done on dialogue systems [Williams and Young, 2007], research on NLG systems should also in the future make more realistic assumptions about their state of knowledge and address the challenges arising with it.

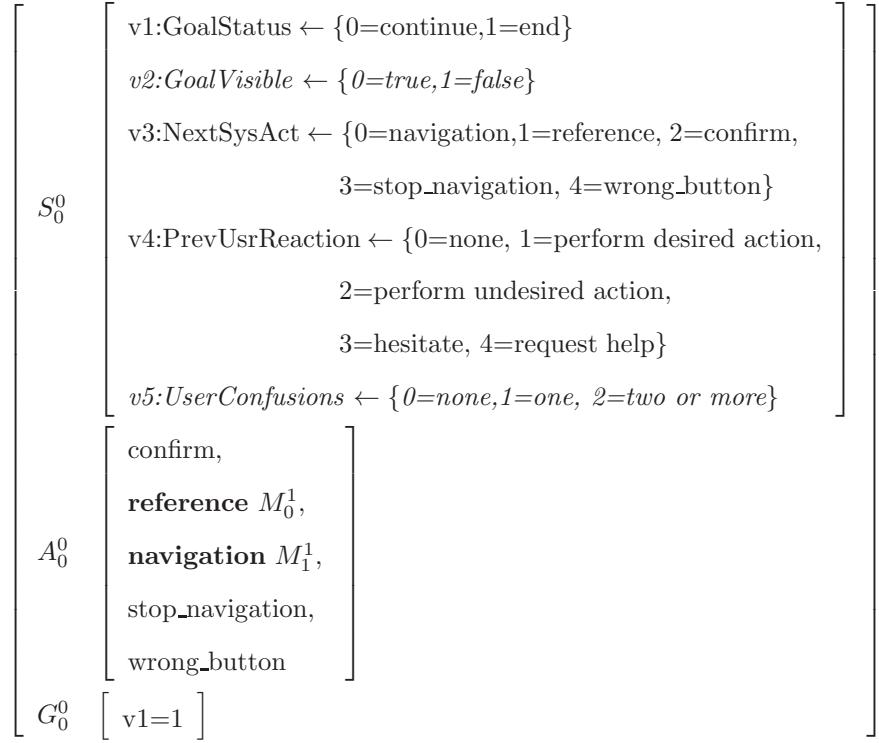
Transfer to New Domains The work in this thesis has been applied to a specific domain, the generation of route instructions and referring expressions in the GIVE scenario, a game task in a virtual 3D world. We have then tested the domain transferability of the learnt policies by transferring it to the task of generating navigation instructions in a real in-situ navigation task. To evaluate the usefulness of the suggested methods on a larger scale, however, and in less similar settings than done so far, we could apply hierarchical reinforcement learning with graphical models to new domains, such as text or paragraph generation. In addition, an application to other NLP tasks is conceivable, such as sentence compression, summarisation, machine translation or human-robot interaction.

Appendix A

To develop a complete NLG system that can generate flexible and adaptive instructions for the GIVE task, we will need learning agents for the tasks of low-level, high-level and mixed navigation, for reference generation and for troubleshooting with respect to content selection. Further, we will need to deal with utterance planning and surface realisation of different instruction types. Figure 3.9 shows a hierarchy of learning agents for these tasks. We see a root agent on the top which allocates different tasks to its children. We then have a left-hand branch dealing with reference phenomena and a right-hand branch dealing with navigation phenomena. In the middle, we find agents for utterance planning and repair that are shared by both the reference and the navigation branch. Finally, we find agents for surface realisation in the bottom of the hierarchy. Let us now go through all agents of the hierarchy individually and inspect their state and action sets. We will also see those state variables that are shared among agents of different levels to allow a joint optimisation of different behaviours. Each agent is described using a feature structure consisting of its corresponding states S ,

7. APPENDIX A

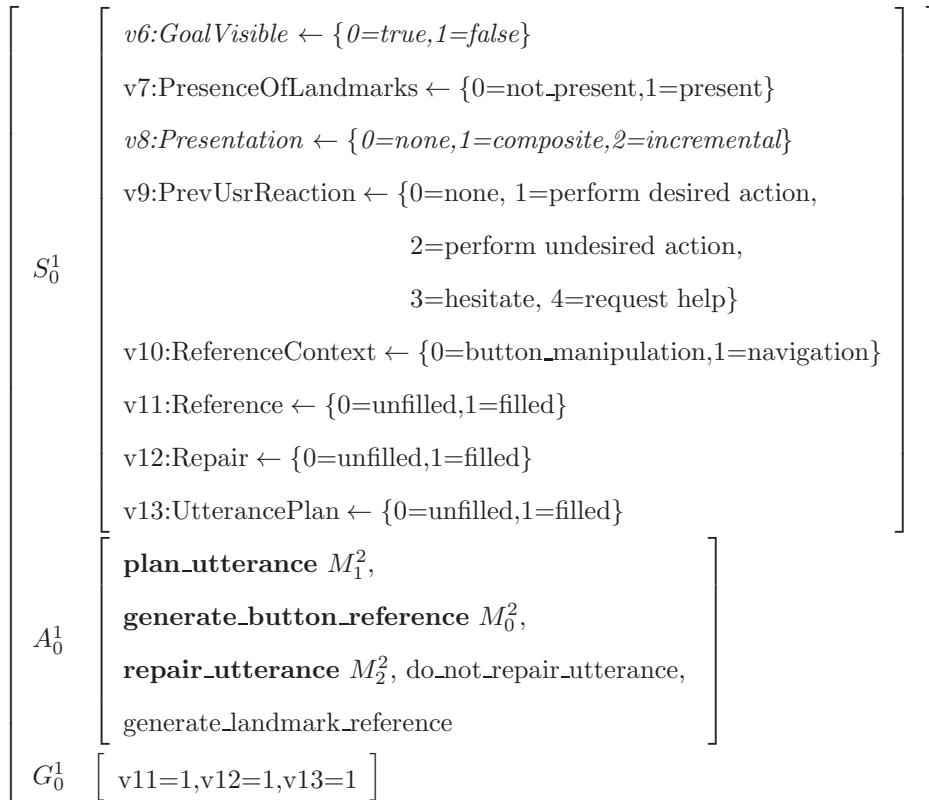
actions A and goal states $G \in S$.



Agent M_0^0 is the root agent. It chooses the next system action based on the user's degree of confusion and their reaction to the last system utterance. It also knows whether the next goal is already visible or not. Both 'GoalVisible' and 'UserConfusion' are variables shared with other agents. All shared variables are given in cursive fonts. The agent can choose to confirm the user's last action (such as saying 'very good'), to tell the user to stop walking or that an unintended button referent was pressed. These are primitive actions. Alternatively, the root agent can decide to generate a navigation or a reference instruction. In this case, a composite action is called and control is passed down either to agent M_0^1 (reference) or agent M_1^1 (navigation). Composite actions are indicated in bold-face. The root reaches its goal state G_0^0 and terminates when all instructions have been

successfully generated and an interaction is over. The environment will then set the ‘GoalStatus’ variable to 1.

Whenever a reference to an object is needed, agent M_0^1 will be called. It analyses the context of the reference to decide whether a reference to a button is intended or to a salient landmark that is located within the environment. In addition, it again has knowledge of whether the next goal is visible and what type of presentation has been chosen in utterance planning (if any, yet).



In case of a button reference, a subtask¹ is called, the button reference agent

¹The terms agent and subtask are used interchangeably.

7. APPENDIX A

M_0^2 . In case of a landmark reference, a primitive action is executed.

S_0^2	<pre> v24:ColourDistractor ← {0=unfilled,1=filled} v25:ColourReferent ← {0=unfilled,1=filled} v26:DiscriminatingColourDistractor ← {0=false,1=true} v27:DiscriminatingColourReferent ← {0=false,1=true} v28:Distractor ← {0=unfilled,1=filled} v29:HorizontalRow ← {0=false,1=true} v30:Horizontal ← {0=unfilled,1=filled} v31:NumberOfDistractors ← {0=none,1=one, 2=two or more} v32:PositionInConfiguration ← {0=corner,1=edge,2=middle, 3=only_button,4=other} v33:Position ← {0=unfilled,1=filled} v34:Surface ← {0=unfilled,1=filled} v35:Type ← {0=unfilled,1=filled} v36:VerticalRow ← {0=false,1=true} v37:Vertical ← {0=unfilled,1=filled} </pre>
A_0^2	<p>referring_expression M_0^3,</p> <pre> include_distractor, do_not_include_distractor, include_type, do_not_include_type, include_referent_colour, do_not_include_referent_colour, include_distractor_colour, do_not_include_distractor_colour, include_horizontal_position, do_not_include_horizontal_position, include_vertical_position, do_not_include_vertical_position, include_position_in_configuration </pre>
G_0^2	$\left[\begin{array}{l} v24=1, v25=1, v28=1, v30=1, v33=1, v34=1, v35=1, v37=1 \end{array} \right]$

Moreover, the reference agent has to make an utterance plan (by calling sub-task M_1^2) and it has to observe the user's last reaction based on which it can

either decide to repair an utterance (that was unsuccessful) or generate the next. The goal state G_0^1 is reached when these three decisions, on the type of reference, the utterance plan and the repair, have been made. Note the special feature ‘unfilled/filled’ that is used in the state representation several times. It indicates whether a decision has been made at this point. Assuming that a reference to a button is required, control will be passed to agent M_0^2 , where all important content selection decisions for referring expressions are made. The agent decides whether to include a button’s colour, its position, its distractors and the distractors’ colours. All of these decisions can be based on properties of the environment, such as whether the referent or its distractors have discriminating colours, or whether the referent’s spatial position, its ‘PositionInConfiguration’ is likely to provide a unique identification. When all content selection decisions have been made, a surface form needs to be generated. To this end, control is passed to the agent’s child, the referring expression generation leaf agent M_0^3 . Once control will be transferred back, agent M_0^2 has reached its goal state G_0^2 .

Agent M_0^3 needs to make lexical choices for determiners, (button) types and a verb. In addition, it needs to decide whether to realise a spatial relation as an adverbial phrase (as in ‘left button’), as a prepositional phrase (as in ‘button on the left’) or as a relative clause with a prepositional phrase (as in ‘button that is on the left’). For all lexical and syntactic choices, there is always also an option of leaving the element empty, i.e. eliding the constituent. The agent reaches its goal state G_0^3 when a surface form has been chosen. This is also indicated by the negations (\neg) in the goal state that ensure that a decision should have been made for each constituent before termination. It can use the content selection choices

7. APPENDIX A

made by its parent agent to motivate its own decisions.

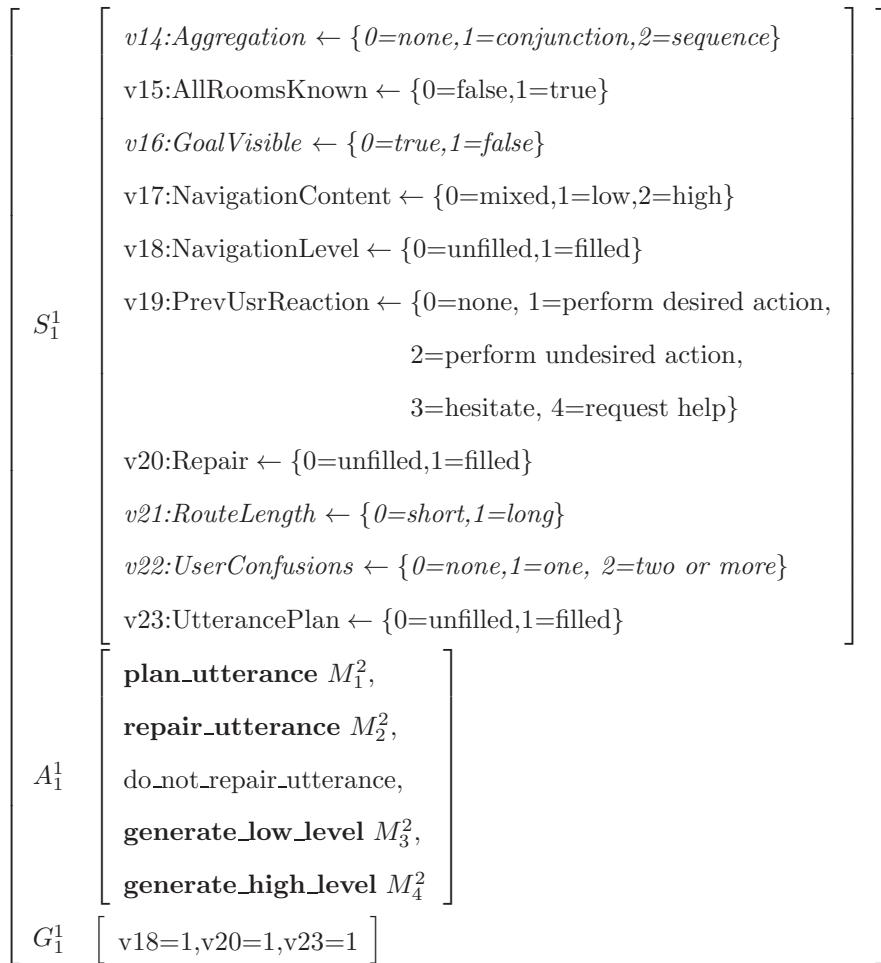
S_0^3	$v66:Distractor \leftarrow \{0=true,1=false\}$ $v67:Landmark \leftarrow \{0=true,1=false\}$ $v68:Position \leftarrow \{0=true,1=false\}$ $v69:ReDeterminer \leftarrow \{0=unfilled,1=the,2=that,3=empty\}$ $v70:ReSpatialRelation \leftarrow \{0=unfilled,1=adv,2=pp,3=rel_clause_pp\}$ $v71:ReType \leftarrow \{0=unfilled,1=button, 2=one, 3=it, 4=empty\}$ $v72:ReVerb \leftarrow \{0=unfilled,1=push,2=press,3=click,4=click_on,5=choose,$ $6=get,7=hit, 8=empty\}$ $v73:UserConfusions \leftarrow \{0=none,1=one, 2=two \text{ or } more\}$ $\text{det_the, det_that, det_empty,}$ $\text{sr_adv, sr_pp, sr_rel_clause_pp,}$
A_0^3	$\text{type_button, type_one, type_empty, type_it,}$ $\text{verb_push, verb_press, verb_click, verb_click_on,}$ $\text{verb_choose, verb_get, verb_hit, verb_empty}$
G_0^3	$\left[v69 \neg 0, v70 \neg 0, v71 \neg 0, v72 \neg 0 \right]$

After having covered the left-most branch of the hierarchy of learning agents, let us now investigate the scenario of generating navigation instructions.

We assume that the root agent M_0^0 has decided to generate a navigation instruction and control has been passed to agent M_1^1 . The decisions we are facing here include whether to use a high-level navigation strategy (and call subtask M_4^2) or a low-level navigation strategy (and call subtask M_3^2). Mixed strategies appear by alternating these two choices. Furthermore, the agent needs to generate an utterance plan by calling subtask M_1^2 , and it needs to decide whether to repair

a previous utterance (when the observed user reaction was negative) or to not repair and instead generate the next instruction.

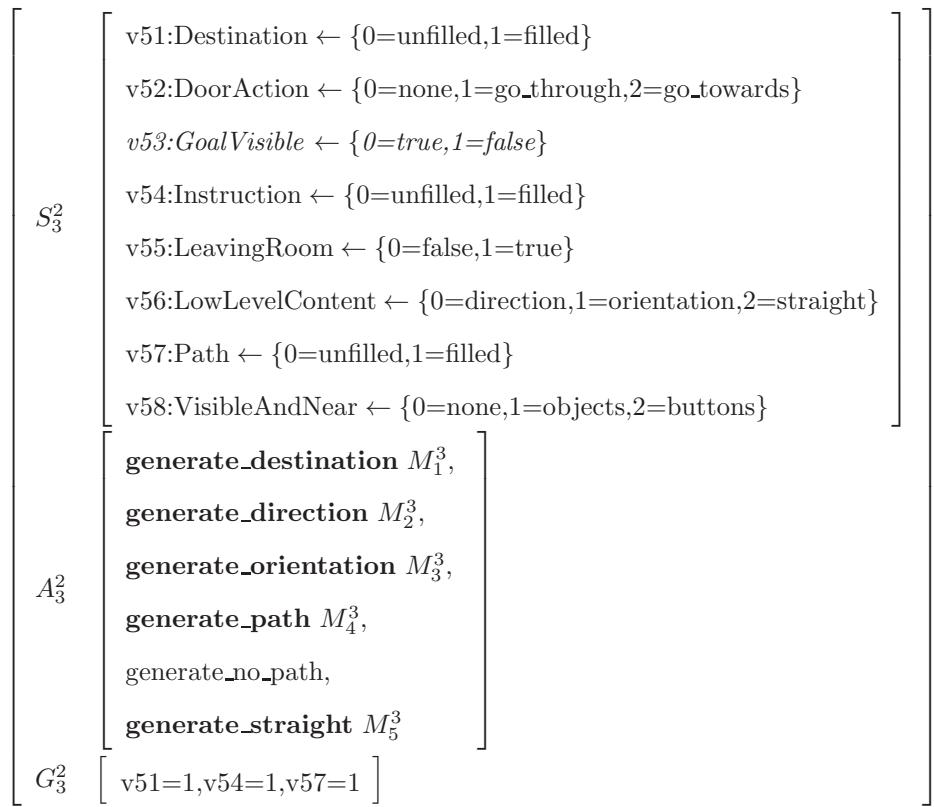
These decisions can also be based on the user’s degree of confusion, certain utterance planning decisions or the length of a particular route. The goal state G_1^1 is reached when they have been made.



Let us assume that agent M_1^1 chose to generate a low-level instruction and to pass control to agent M_3^2 . At this level, we can decide to generate a direction instruction, an orientation instruction or a ‘straight’ instruction. In either case, one of the agents M_1^3 , M_2^3 or M_5^3 will be called.

7. APPENDIX A

Which of these is suitable depends on the environment and the user's next action. In addition to one of these, the agent can decide to provide additional (optional) information to the user regarding the path they should take (by calling agent M_4^3) or the destination they should head towards (by calling agent M_1^3). Several environmental properties, such as whether the user is leaving the room, the next goal is visible or the destination is visible and near, aid the agent's decision making until the goal state G_3^2 is reached.



Alternatively, the parent navigation agent could have chosen to generate a high-level navigation instruction. Agent M_4^2 is responsible for these. It can then either call agent M_1^3 in order to generate a destination instruction or call agent

M_4^3 to generate a path instruction.

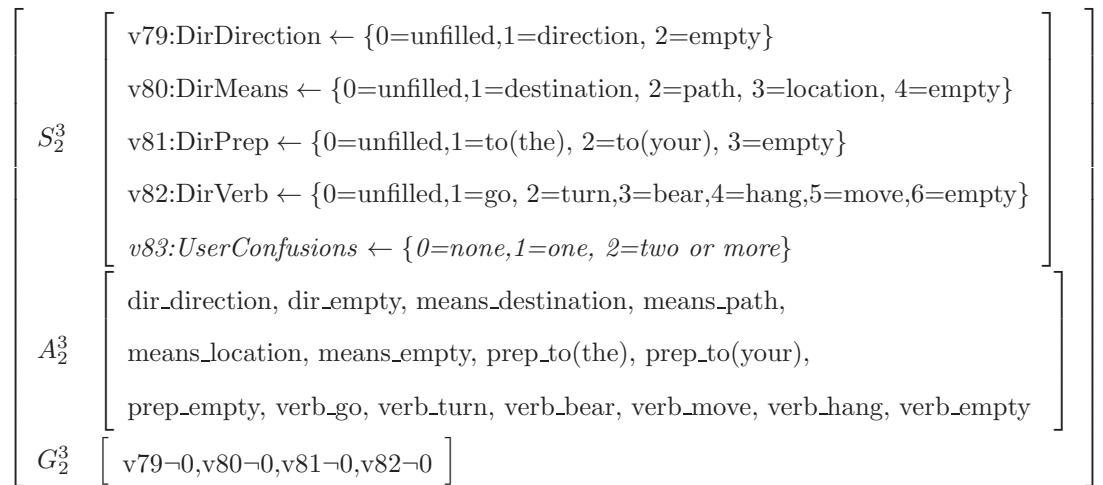
S_4^2	v59:DestinationType $\leftarrow \{0=\text{other}, 1=\text{landmark}, 2=\text{button}\}$ <i>v60:GoalVisible</i> $\leftarrow \{0=\text{true}, 1=\text{false}\}$ v61:Instruction $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v62:LeavingRoom $\leftarrow \{0=\text{false}, 1=\text{true}\}$ v63:NextRoomEquals $\leftarrow \{0=\text{same}, 1=\text{previous}, 2=\text{corridor}, 3=\text{other}\}$ v64:Path $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v65:Surface $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$
A_4^2	referring_expression M_0^3 , generate_destination M_1^3 , generate_path M_4^3 , generate_no_path
G_4^2	$\left[\begin{array}{l} v61=1, v64=1, v65=1 \end{array} \right]$

S_1^3	v74:DesDirection $\leftarrow \{0=\text{unfilled}, 1=\text{direction}, 2=\text{straight}, 3=\text{empty}\}$ v75:DesPrep $\leftarrow \{0=\text{unfilled}, 1=\text{to}, 2=\text{towards}, 3=\text{empty}, 4=\text{into}, 5=\text{in}, 6=\text{until}\}$ v76:DesRelatum $\leftarrow \{0=\text{unfilled}, 1=\text{room}, 2=\text{landmark}, 3=\text{empty}\}$ v77:DesVerb $\leftarrow \{0=\text{unfilled}, 1=\text{go}, 2=\text{keep_going}, 3=\text{empty}, 4=\text{get}, 5=\text{return}, 6=\text{continue}, 7=\text{walk}\}$ <i>v78:UserConfusions</i> $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$
A_1^3	dir_direction, dir_straight, dir_empty, prep_to, prep_towards, prep_empty, prep_into, prep_in, prep_until, relatum_room, relatum_landmark, relatum_empty, verb_continue, verb_go, verb_walk, verb_get, verb_return, verb_empty, verb_keep-going
G_1^3	$\left[\begin{array}{l} v74=0, v75=0, v76=0, v77=0 \end{array} \right]$

If a destination instruction is needed, the referring expression agent M_0^3 can be

7. APPENDIX A

called in addition, if the target destination is a button. If the goal is a landmark, the referring expression is not needed. The goal state G_4^2 is reached when a complete instruction has been generated. Whatever type of navigation instruction was chosen at earlier levels, one or more of the navigation leaf agents $M_1^3 \dots M_5^3$ will have to be consulted for a surface form. Destination instructions, such as ‘go back to the room with the trophy’, are realised by agent M_1^3 . It needs to choose a lexical realisation for the verb of the instruction, a preposition, a relatum and optional further detail which can consist of giving a direction or telling the user to go straight. Again, all constituents can also be realised by an elided surface form and the agent has the possibility to base decisions on the user’s degree of confusion. The goal state G_1^3 has been reached when a complete destination instruction was generated.



Direction instructions are realised by agent M_2^3 . An example of a direction instruction is ‘turn left at the door’. The relevant decisions here are to choose a

verb, a preposition, a direction and optional further information, i.e. the means (a location, e.g). The agent has again information on how confused the user is and can choose to elide any piece of information as long as a grammatical surface form is realised. When this condition is met, the goal state G_2^3 has been reached.

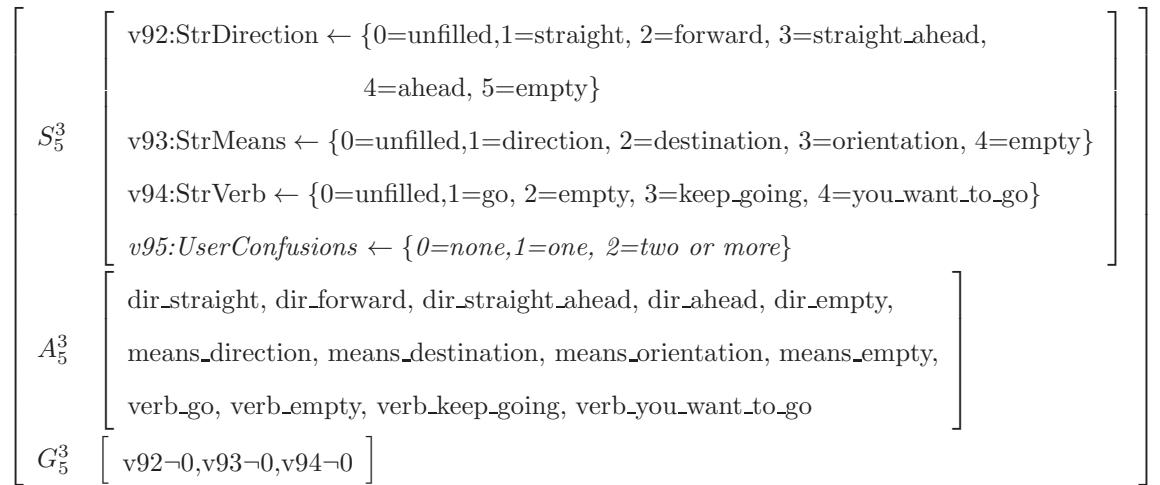
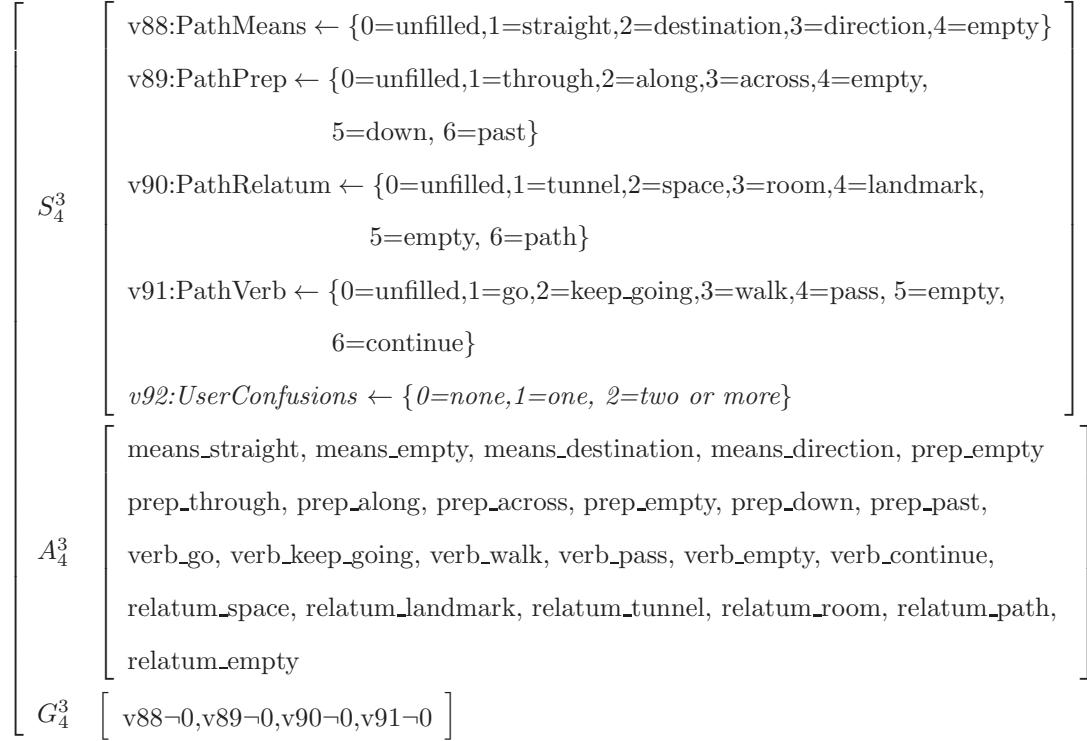
Agent M_3^3 is responsible for generating orientation instructions such as ‘turn 90 degrees to the left’. The process of realising a full surface form involves the decisions of choosing a verb, a direction and an optional means for the instruction. The goal state G_3^3 is reached when these decisions have been made.

$$\begin{array}{c}
 S_3^3 \\
 \left[\begin{array}{l} v84:\text{Direction} \leftarrow \{0=\text{unfilled}, 1=\text{around}, 2=\text{round}, 3=\text{degrees}, 4=\text{empty}\} \\ v85:\text{OriMeans} \leftarrow \{0=\text{unfilled}, 1=\text{path}, 2=\text{destination}, 3=\text{empty}\} \\ v86:\text{OriVerb} \leftarrow \{0=\text{unfilled}, 1=\text{turn}, 2=\text{you_want_to_turn}\} \\ v87:\text{UserConfusions} \leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\} \end{array} \right] \\
 A_3^3 \\
 \left[\begin{array}{l} \text{dir_around}, \text{dir_round}, \text{dir_degrees}, \text{dir_empty}, \text{means_path}, \text{means_empty}, \\ \text{means_destination}, \text{verb_turn}, \text{verb_you_want_to_turn} \end{array} \right] \\
 G_3^3 \\
 \left[\begin{array}{l} v84 \neg 0, v85 \neg 0, v86 \neg 0 \end{array} \right]
 \end{array}$$

To generate path instructions such as ‘Walk past the copy room’, agent M_4^3 needs to be consulted. It will choose a verb for the instruction, a preposition, a relatum and an optional means constituent depending on the user’s degree of confusion. When an instruction has been fully realised, the agent terminates,

7. APPENDIX A

reaches goal state G_4^3 , and passes control back to its parent.



Finally, we can call agent M_5^3 to generate a ‘straight’ instruction, e.g. ‘Go

straight'. To do that and reach the goal state, the agent needs to choose lexical items for a verb and a direction and whether or not to include a piece of optional information. If optional information should be included, it can either be the direction of heading (after going straight), the destination of going straight, or an orientation (before going straight). The agent reaches its goal state G_5^3 when a full instruction was realised. Control is then passed back to the parent, where the parent's state representation will be updated to reflect the completed surface realisation task.



Apart from content selection and surface realisation, we have seen that for all

7. APPENDIX A

navigation and reference instructions, utterance planning needs to be consulted to decide on how to present the generated instruction. Agent M_1^2 is responsible for this.

It faces the decisions of whether or not to aggregate instructions by a conjoining conjunction ‘and’ or a sequential conjunction ‘then’ and what type of information structure to choose, marked, e.g. in placing a temporal marker in thematic position, or unmarked, i.e. placing the grammatical subject in thematic position. Further, instructions can be presented to the user in a composite fashion, that is all together, or incrementally, that is one after the other. The latter method may ease the user’s comprehension process when there is a large number of instructions. Finally, temporal markers (e.g. ‘first’, ‘now’, ‘finally’) can be included in a set of utterances to impose an explicit sequential structure. Again, the agent reaches its goal state G_1^2 when these decisions have been made.

S_2^2	$v47:GoalVisible \leftarrow \{0=true, 1=false\}$
	$v48:NavigationLevelContent \leftarrow \{0=low_level, 1=high_level\}$
	$v49:Repair \leftarrow \{0=unfilled, 1=filled\}$
	$v50:UserConfusions \leftarrow \{0=none, 1=one, 2=two \text{ or } more\}$
A_2^2	paraphrase_utterance, repeat_utterance, switch_navigation_level
G_2^2	$v40=1$

Finally, previously unsuccessful utterances can be repaired by calling agent M_2^2 . It generally has the options of either repeating a previous utterance or rephrasing it. When a navigation instruction is repaired, the agent has the additional option of switching the navigation strategy, for example, from high to low,

or vice versa. The repair agent terminates by reaching goal state G_2^2 when one of these decisions has been made.

As discussed in Chapter 3, the hierarchical agent we designed has a state-action space of $|S \times A| = \sum_i |f_i| \times |A| = 1,480,869$. In contrast, a flat reinforcement learning agent using the same states and actions would have a state-action space of $|S \times A| = \sum_i |f_i| \times |A| = 3 \times 10^{57}$, which indicates the advantage of using a hierarchical decomposition. For a formal discussion of the computational complexity of solving MDPs, please see Littman et al. [1995]; Papadimitriou and Tsitsiklis [1987].

7. APPENDIX A

Appendix B

In Chapter 3, Section 3.4.4, we defined a hierarchy of learning agents for instruction and referring expression generation in situated interaction. Let us now augment this hierarchy with a hierarchical information state specifying constraints on the actions available in each generation state of the hierarchy. Note that the information state does not require variables of different subtasks to be shared for a jointly optimised behaviour. The joint optimisation is dealt with exclusively by the learning agent. Again, composite actions are given in bold-face fonts as well as the preconditions (**pre**) and effects (**eff**) of the information state. Note that preconditions use logical notations such $=\!=$ for equality, \wedge for conjunction, \vee for

7. APPENDIX B

disjunction and \neg for negation.

IS_0^0	<pre> v1:GoalStatus ← {0=continue,1=end} v2:NextSysAct ← {0=navigation,1=reference, 2=confirm, 3=stop_navigation, 4=wrong_button} v3:NextSysActValue ← string v4:PrevUsrReaction ← {0=none, 1=perform desired action, 2=perform undesired action, 3=wait, 4=request help} v5:UserConfusions ← {0=none,1=one, 2=two or more} </pre>
$URules_0^0$	<pre> confirm pre: v2==2 ∧ ¬v3=='confirm' eff: v3 ← 'confirm' reference M_0^1 pre: v2==1 ∧ ¬v3=='reference' eff: v3 ← 'reference' navigation M_1^1 pre: v2==0 ∧ ¬v3=='navigation' eff: v3 ← 'navigation' stop_navigation pre: v2==3 ∧ ¬v3=='stop_navigation' eff: v3 ← 'stop_navigation' wrong_button pre: v2==4 ∧ ¬v3=='wrong_button' eff: v3 ← 'wrong_button' </pre>

The information state of the root agent M_0^0 contains two types of information. First, it contains state variables that indicate the status of the interaction ('GoalStatus'), the next system act, previous user reaction and the user's degree of confusion. These state variables are shared with the state representation of the learning agent defined earlier. In addition, it contains a set of variables that encode the type of decision that was made in order to fill a slot in the information state. The first type of information is formalised using integers (as in the

learning agent), the second type of information is formalised using strings. The update rules of the information state apply to both types of information and are named after the action available to the agent. For example, the action ‘confirm’ is available under the precondition that the variable v2 equals 2 and the variable v3 does not equal ‘confirm’. Taking the action ‘confirm’ then has the effect that variable v3 is set to reflect the action chosen, i.e. ‘confirm’.

IS_0^1	<pre> v6:PresenceOfLandmarks ← {0=not-present,1=present} v7:PrevUsrReaction ← {0=none, 1=perform desired action, 2=perform undesired action, 3=wait, 4=request help} v8:ReferenceContext ← {0=button_manipulation,1=navigation} v9:Reference ← {0=unfilled,1=filled} v10:ReferenceValue ← <i>string</i> v11:Repair ← {0=unfilled,1=filled} v12:RepairValue ← <i>string</i> v13:UtterancePlan ← {0=unfilled,1=filled} plan_utterance M_1^2 pre: v13==0 eff: v13 ← 1 generate_button_reference M_0^2 pre: v9==0 ∧ v6==0 eff: v9 ← 1, v10 ← ‘button_reference’ </pre>
$URules_0^1$	<pre> repair_utterance M_2^2 pre: v11==0 eff: v11 ← 1, v12 ← ‘repair’ do_not_repair_utterance pre: v11==0 eff: v11 ← 1, v12 ← ‘do_not_repair’ generate_landmark_reference pre: v9==0 ∧ v6==1 eff: v9 ← 1, v10 ← ‘landmark_reference’ </pre>

7. APPENDIX B

Note that update rules are defined in a way that makes several actions be available at the same time. The purpose of this liberal definition is that the best sequence of actions is not clear and should therefore be subject to optimisation of the learning agent. On the other hand, the definition has the effect of constraining actions in a way that decision points are not considered twice. For instance, once a decision has been made on whether to generate a reference instruction, a navigation instruction, a confirmation, stop instruction or button warning, variable v3 will be updated to the fact that the action has been taken and does therefore not have to be considered anymore. The advantage is that the agent's search space is reduced to those actions that can still cause changes in the environment.

Model M_0^1 is responsible for generating reference instructions. Its information state is organised in a similar fashion as in the root agent. We distinguish two types of information, state variables shared with the learning agent and state variables that encode decisions. The former contain information on the presence of landmarks, the previous user reaction, the context of the reference (reference to a button or to a landmark), the utterance plan and whether the utterance is repairing a previous utterance or not. The second type of information is updated as actions are taken. The update rules also resemble the ones specified for the root agent, but they contain a number of extra constraints. The action 'generate_button_reference' is only valid when buttons are present, and the action 'generate_landmark_reference' is only valid when landmarks are present. Again, decision points can only be considered once per instruction. When for example a navigation level has been chosen for the current utterance, this decision can only be revised in the next instruction.

IS_1^1	<pre> v35:AllRoomsKnown ← {0=false,1=true} v36:NavigationContent ← {0=mixed,1=low,2=high} v37:NavigationLevel ← {0=unfilled,1=filled} v38:NavigationLevelValue ← <i>string</i> v39:PrevUsrReaction ← {0=none, 1=perform desired action, 2=perform undesired action, 3=wait, 4=request help} v40:Repair ← {0=unfilled,1=filled} v41:RepairValue ← <i>string</i> v42:UserConfusions ← {0=none,1=one,2=two or more} v43:UtterancePlan ← {0=unfilled,1=filled} </pre>
$URules_1^1$	<pre> plan_utterance M_1^2 pre: v43==0 eff: v43 ← 1 repair_utterance M_2^2 pre: v40==0 eff: v40 ← 1, v41 ← ‘repair’ do_not_repair_utterance pre: v40==0 eff: v40 ← 1, v41 ← ‘do_not_repair’ generate_low_level M_3^2 pre: v37==0 ∨ (v37==0 ∧ v36==1) eff: v37 ← 1, v38 ← ‘low_level’ generate_high_level M_4^2 pre: v37==0 ∨ (v37==0 ∧ v36==2) eff: v37 ← 1, v38 ← ‘high_level’ </pre>

The navigation agent M_1^1 follows the same principle as the previous agents. Its information state contains information shared with the learning agent. It holds information on the rooms that the user knows, the content of the navigation instruction, the navigation level, the status of the utterance plan, the previous user reaction, and whether or not the current utterance is a repair of the previous

7. APPENDIX B

one. In addition, it contains a set of variables recording all actions that have been taken. There are two ways of generating a low-level or a high-level navigation action. The decision can either be left to the learning agent, or it can be pre-specified by setting variable v36 to 1 for low-level navigation or to 2 for high-level navigation.

IS_0^2	<p>v14:ColourDistractor $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v15:ColourDistractorValue $\leftarrow \text{string}$ v16:ColourReferent $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v17:ColourReferentValue $\leftarrow \text{string}$ v18:DiscriminatingColourDistractor $\leftarrow \{0=\text{false}, 1=\text{true}\}$ v19:DiscriminatingColourReferent $\leftarrow \{0=\text{false}, 1=\text{true}\}$ v20:Distractor $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v21:DistractorValue $\leftarrow \text{string}$ v22:HorizontalRow $\leftarrow \{0=\text{false}, 1=\text{true}\}$ v23:Horizontal $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v24:HorizontalValue $\leftarrow \text{string}$ v25:NumberOfDistractors $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$ v26:PositionInConfiguration $\leftarrow \{0=\text{corner}, 1=\text{edge}, 2=\text{middle},$ $3=\text{only_button}, 4=\text{other}\}$ v27:Position $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v28:PositionValue $\leftarrow \text{string}$ v29:Surface $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v30>Type $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v31>TypeValue $\leftarrow \text{string}$ v32:VerticalRow $\leftarrow \{0=\text{false}, 1=\text{true}\}$ v33:Vertical $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$ v34:VerticalValue $\leftarrow \text{string}$</p>
----------	---

$URules_0^2$

referring_expression M_0^3 **pre:** $v29==0$

eff: $v29 \leftarrow 1$

include_distractor **pre:** $v20==0 \wedge \neg v25==0 \wedge v32==1$

eff: $v20 \leftarrow 1, v21 \leftarrow \text{'use_distractor'}$

do_not_include_distractor **pre:** $v20==0$

eff: $v20 \leftarrow 1, v21 \leftarrow \text{'do_not_use_distractor'}$

include_type **pre:** $v30==0$

eff: $v30 \leftarrow 1, v31 \leftarrow \text{'include_type'}$

do_not_include_type **pre:** $v30==0$

eff: $v30 \leftarrow 1, v31 \leftarrow \text{'do_not_include_type'}$

include_referent_colour **pre:** $v16==0$

eff: $v16 \leftarrow 1, v17 \leftarrow \text{'use_referent_colour'}$

do_not_include_referent_colour **pre:** $v16==0$

eff: $v16 \leftarrow 1, v17 \leftarrow \text{'do_not_use_referent_colour'}$

include_distractor_colour **pre:** $v14==0 \wedge v18==1$

eff: $v14 \leftarrow 1, v15 \leftarrow \text{'use_distractor_colour'}$

do_not_include_distractor_colour **pre:** $v14==0$

eff: $v14 \leftarrow 1, v15 \leftarrow \text{'do_not_use_distractor_colour'}$

include_horizontal_position **pre:** $v23==0 \wedge v22==1$

eff: $v23 \leftarrow 1, v24 \leftarrow \text{'use_horizontal'}$

do_not_include_horizontal_position **pre:** $v23==0$

eff: $v23 \leftarrow 1, v24 \leftarrow \text{'do_not_use_horizontal'}$

include_vertical_position **pre:** $v33==0 \wedge v32==1$

eff: $v33 \leftarrow 1, v34 \leftarrow \text{'use_vertical'}$

do_not_include_vertical_position **pre:** $v33==0$

eff: $v33 \leftarrow 1, v34 \leftarrow \text{'do_not_use_vertical'}$

include_position_in_configuration **pre:** $27==0$

eff: $v27 \leftarrow 1, v28 \leftarrow \text{'use_position'}$

7. APPENDIX B

As before, alternating these choices will result in a mixed instruction giving strategy. All actions concerning the same decision point are only available to the agent once per instruction.

Model M_0^2 is responsible for content selection of referring expressions referring to buttons. In the information state we encode information on the presence of distractors and landmarks, on whether or not referent and distractors have discriminating colours, the position of the referent in relation to its distractors and the type of semantic decisions that are available or have already been made. As with all other agents, the information state allows decision points to be considered only once per instruction to prevent the agent from entering into loops. Additionally, the information state keeps a record of all decisions that have been made. The update rules state that distractors can only be used when salient distractors are present and the colour of a distractor should always be used when it is discriminating. Note that the colour of a distractor can always be used, but it is only obligatory when the colour is discriminating.

The information state of model M_1^2 , which is responsible for the utterance plan of instructions, holds information on the user's previous reaction and their degree of confusion. Furthermore, it contains information on the semantic choices made for the instruction, i.e. the type of aggregation, information structure, way of presentation and the inclusion of temporal markers. It keeps a record of all decisions made and allows decision points to be considered only once per

instruction.

IS_1^2	<pre> v44:Aggregation ← {0=unfilled,1=filled} v45:AggregationValue ← <i>string</i> v46:InfoStructure ← {0=unfilled,1=filled} v47:InfoStructureValue ← <i>string</i> v48:NumberOfInstructions ← {0=none,1=one, 2=two,3=three or more} v49:Presentation ← {0=unfilled,1=filled} v50:PresentationValue ← <i>string</i> v51:PrevUsrReaction ← {0=none, 1=perform desired action, 2=perform undesired action, 3=wait, 4=request help} v52:TemporalMarkers ← {0=unfilled,1=filled} v53:TemporalMarkersValue ← <i>string</i> v54:UserConfusions ← {0=none,1=one, 2=two ore more} choose_aggregation pre: v44==0 eff: v44 ← 1, v45 ← ‘aggregate’ choose_no_aggregation pre: v44==0 eff: v44 ← 1, v45 ← ‘no_aggregate’ choose_temporal_markers pre: v52==0 eff: v52 ← 1, v53 ← ‘use_temp_markers’ choose_no_temporal_markers pre: v52==0 eff: v52 ← 1, v53 ← ‘use_no_temp_markers’ choose_marked_theme pre: v46==0 eff: v46 ← 1, v47 ← ‘marked_theme’ choose_unmarked_theme pre: v46==0 eff: v46 ← 1, v47 ← ‘unmarked_theme’ choose_joint_presentation pre: v49==0 eff: v49 ← 1, v50 ← ‘joint_presentation’ choose_incremental_presentation pre: v49==0 eff: v49 ← 1, v50 ← ‘incremental_presentation’ </pre>
$URules_1^2$	

7. APPENDIX B

Whenever a previous utterance was unsuccessful and needs to be repaired, model M_2^2 is called. It contains information on the navigation level and the user's degree of confusion. The update rules specify no constraints other than that decision points should not be considered twice. The repair agent has the options of paraphrasing an unsuccessful utterance, repeating it or, when generating a navigation instruction, to switch the navigation level and generate the utterance again.

IS_2^2	<table border="0"> <tr> <td>v55:NavigationLevelContent $\leftarrow \{0=\text{low_level}, 1=\text{high_level}\}$</td></tr> <tr> <td>v56:Repair $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$</td></tr> <tr> <td>v57:RepairValue $\leftarrow \text{string}$</td></tr> <tr> <td>v58:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$</td></tr> </table>	v55:NavigationLevelContent $\leftarrow \{0=\text{low_level}, 1=\text{high_level}\}$	v56:Repair $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$	v57:RepairValue $\leftarrow \text{string}$	v58:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$
v55:NavigationLevelContent $\leftarrow \{0=\text{low_level}, 1=\text{high_level}\}$					
v56:Repair $\leftarrow \{0=\text{unfilled}, 1=\text{filled}\}$					
v57:RepairValue $\leftarrow \text{string}$					
v58:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$					
$URules_2^2$	<table border="0"> <tr> <td>paraphrase_utterance pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'paraphrase'}$</td></tr> <tr> <td>repeat_utterance pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'repeat'}$</td></tr> <tr> <td>switch_navigation_level pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'switch_navigation_level'}$</td></tr> </table>	paraphrase_utterance pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'paraphrase'}$	repeat_utterance pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'repeat'}$	switch_navigation_level pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'switch_navigation_level'}$	
paraphrase_utterance pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'paraphrase'}$					
repeat_utterance pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'repeat'}$					
switch_navigation_level pre: v56==0 eff: v56 $\leftarrow 1$, v57 $\leftarrow \text{'switch_navigation_level'}$					

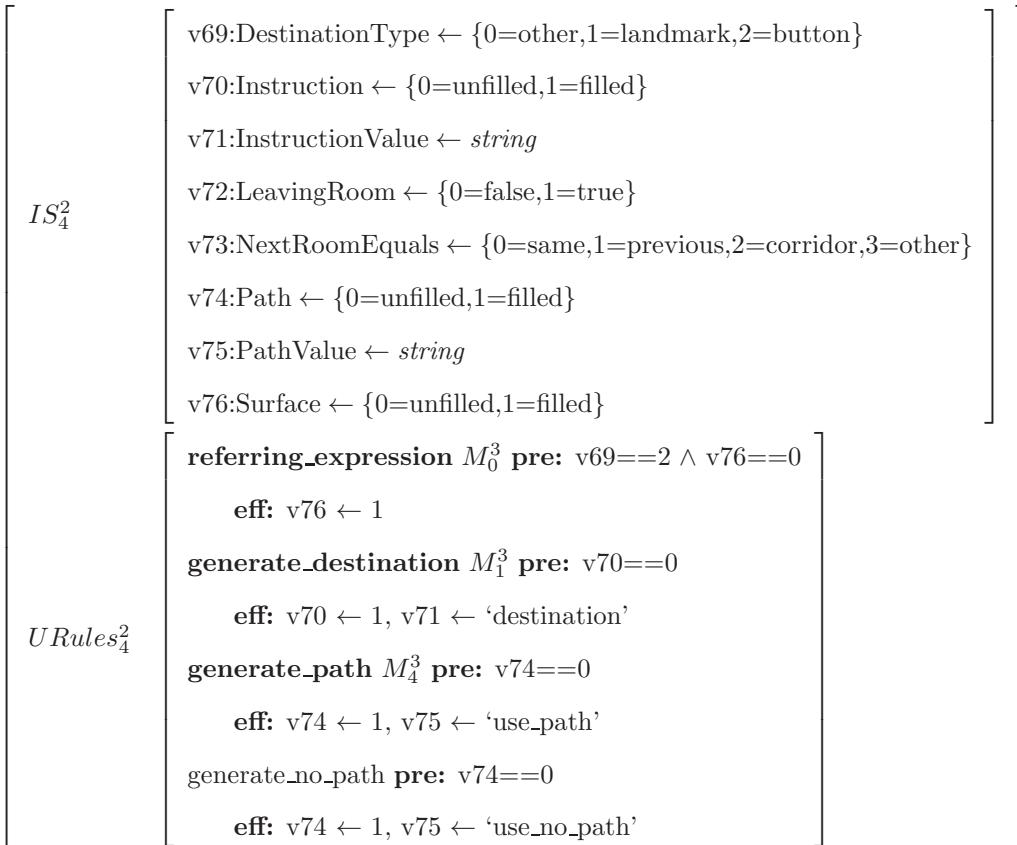
As previously discussed, navigation instructions can be low-level or high-level. Model M_3^2 generates the former type. Its information state holds information on whether the user is leaving the room, needs to go towards or through a door, and whether buttons or other objects are visible and near. It also keeps a record of the decisions it has made. The update rules of the information state constrain action selection as follows. Variable v65 indicates the next (and only valid) instruction to be generated of direction, orientation and 'straight' (which is enforced by a

precondition). This information needs to be set from the spatial environment of an interaction. Further, the update rules state that a destination instruction should be generated additionally if the user needs to approach a door and that a path instruction is needed when the user is required to go through the door.

IS_3^2	<pre> v59:Destination ← {0=unfilled,1=filled} v60:DestinationValue ← <i>string</i> v61:DoorAction ← {0=none,1=go_through,2=go_towards} v62:Instruction ← {0=unfilled,1=filled} v63:InstructionValue ← <i>string</i> v64:LeavingRoom ← {0=false,1=true} v65:LowLevelContent ← {0=direction,1=orientation,2=straight} v66:Path ← {0=unfilled,1=filled} v67:PathValue ← <i>string</i> v68:VisibleAndNear ← {0=none,1=objects,2=buttons} generate_destination M_1^3 pre: v59==0 ∧ v61==2 eff: v59 ← 1, v60 ← ‘use_destination’ generate_direction M_2^3 pre: v62==0 ∧ v65==0 eff: v62 ← 1, v63 ← ‘direction’ generate_orientation M_3^3 pre: v62==0 ∧ v65==1 eff: v62 ← 1, v63 ← ‘orientation’ generate_path M_4^3 pre: v66==0 ∧ v61==1 eff: v66 ← 1, v67 ← ‘use_path’ generate_no_path pre: v66==0 ∧ ¬ v61==1 eff: v66 ← 1, v67 ← ‘use_no_path’ generate_straight M_5^3 pre: v62==0 ∧ v65==2 eff: v62 ← 1, v63 ← ‘straight’ </pre>
$URules_3^2$	

7. APPENDIX B

Instead of generating a low-level instruction, a high-level navigation level can be chosen. In this case, model M_4^2 will be called. Its information state holds information about the type of destination the user is heading towards, whether it is a button or a landmark, as well as whether the user is leaving the room and the identity of the next room. It encodes a record of all decisions made for the current instruction, but is (as always) not allowed to consider an individual decision point more than once per instruction. The update rules specify the further constraint that a surface form for a referring expression needs to be obtained only when the user is approaching a button.



Whenever this is the case, agent M_0^3 is called, which is responsible for generating surface forms for referring expressions. It is also the first surface realisation agent in a row of six. The information state for referring expressions encodes mostly information on the decisions that the agent needs to make, i.e. how to express the verb, the determiner and the type of the referent. It also keeps a record of all decisions made and allows each one to be considered only once. No further constraints are specified in the update rules.

IS_0^3	<pre> v77:ReDeterminer ← {0=unfilled,1=the,2=that,3=empty} v78:ReDeterminerValue ← <i>string</i> v79:ReSpatialRelation ← {0=unfilled,1=adv,2=pp,3=rel_clause_pp} v80:ReSpatialRelationValue ← <i>string</i> v81:ReType ← {0=unfilled,1=button, 2=one, 3=it, 4=empty} v82:ReTypeValue ← <i>string</i> v83:ReVerb ← {0=unfilled,1=push,2=press,3=click,4=click_on,5=choose, 6=get,7=hit, 8=empty} v84:ReVerbValue ← <i>string</i> v85:UserConfusions ← {0=none,1=one,2=two or more} </pre>
$URules_0^3$	<pre> det_the pre: v77==0 eff: v77 ← 1, v78 ← ‘the’ det_that pre: v77==0 eff: v77 ← 2, v78 ← ‘that’ det_empty pre: v77==0 eff: v77 ← 3, v78 ← ‘empty’ sr_adv pre: v79==0 eff: v79 ← 1, v80 ← ‘adv’ sr_pp pre: v79==0 eff: v79 ← 2, v80 ← ‘pp’ </pre>

7. APPENDIX B

URules₀³(continued)

sr_rel_clause_pp pre: v79==0 eff: v79 ← 3, v80 ← ‘rel_clause’ type_button pre: v81==0 eff: v81 ← 1, v82 ← ‘button’ type_empty pre: v81==0 eff: v81 ← 4, v82 ← ‘empty’ type_one pre: v81==0 eff: v81 ← 2, v82 ← ‘one’ type_it pre: v81==0 eff: v81 ← 3, v82 ← ‘it’ verb_push pre: v83==0 eff: v83 ← 1, v84 ← ‘push’ verb_press pre: v83==0 eff: v83 ← 2, v84 ← ‘press’ verb_click pre: v83==0 eff: v83 ← 3, v84 ← ‘click’ verb_click_on pre: v83==0 eff: v83 ← 4, v84 ← ‘click_on’ verb_choose pre: v83==0 eff: v83 ← 5, v84 ← ‘choose’ verb_get pre: v83==0 eff: v83 ← 6, v84 ← ‘get’ verb_hit pre: v83==0 eff: v83 ← 7, v84 ← ‘hit’ verb_empty pre: v83==0 eff: v83 ← 8, v84 ← ‘empty’
--

The surface realisation models $M_{1\dots 5}^3$ are all responsible for navigation. Agent M_1^3 generates destination instructions that need to contain a verb, a preposition,

a relatum and an optional means of carrying out the instruction. The agent has additional information on the user's degree of confusion. The update rules contain no further constraints apart from being allowed to consider each decision point only once per instruction.

IS_1^3	<pre> v86:DesDirection ← {0=unfilled,1=direction, 2=straight, 3=empty} v87:DesDirectionValue ← <i>string</i> v88:DesPrep ← {0=unfilled,1=to, 2=towards, 3=empty, 4=into, 5=in, 6=until} v89:DesPrepValue ← <i>string</i> v90:DesRelatum ← {0=unfilled,1=room, 2=landmark, 3=empty} v91:DesRelatumValue ← <i>string</i> v92:DesVerb ← {0=unfilled,1=go, 2=keep-going, 3=empty, 4=get, 5=return, 6=continue, 7=walk} v93:DesVerbValue ← <i>string</i> v94:UserConfusions ← {0=none,1=one, 2=two or more} dir_direction pre: v86==0 eff: v86 ← 1, v87 ← 'direction' dir_straight pre: v86==0 eff: v86 ← 2, v87 ← 'straight' dir_empty pre: v86==0 eff: v86 ← 3, v87 ← 'empty' prep_to pre: v88==0 eff: v88 ← 1, v89 ← 'to' prep_towards pre: v88==0 eff: v88 ← 2, v89 ← 'towards' prep_empty pre: v88==0 eff: v88 ← 3, v89 ← 'empty' </pre>
$URules_1^3$	

7. APPENDIX B

URules₁³(continued)

<pre> prep_into pre: v88==0 eff: v88 ← 4, v89 ← ‘into’ </pre>	<pre> prep_in pre: v88==0 eff: v88 ← 5, v89 ← ‘in’ </pre>
<pre> prep_until pre: v88==0 eff: v88 ← 6, v89 ← ‘until’ </pre>	<pre> relatum_room pre: v90==0 eff: v90 ← 1, v91 ← ‘room’ </pre>
<pre> relatum_landmark pre: v90==0 eff: v90 ← 2, v91 ← ‘landmark’ </pre>	<pre> relatum_empty pre: v90==0 eff: v90 ← 3, v91 ← ‘empty’ </pre>
<pre> verb_go pre: v92==0 eff: v92 ← 1, v93 ← ‘go’ </pre>	<pre> verb_keep_going pre: v92==0 eff: v92 ← 2, v93 ← ‘keep-going’ </pre>
<pre> verb_empty pre: v92==0 eff: v92 ← 3, v93 ← ‘empty’ </pre>	<pre> verb_get pre: v92==0 eff: v92 ← 4, v93 ← ‘get’ </pre>
<pre> verb_return pre: v92==0 eff: v92 ← 5, v93 ← ‘return’ </pre>	<pre> verb_continue pre: v92==0 eff: v92 ← 6, v93 ← ‘continue’ </pre>
<pre> verb_walk pre: v92==0 eff: v92 ← 7, v93 ← ‘walk’ </pre>	

Agent M_2^3 generates direction instructions containing a verb, a preposition, a direction and an optional means constituent. The information state holds infor-

mation on which decisions have already been made and which are still open. The update rules only restrict decisions to one per instruction.

IS_2^3

```

v95:DirDirection ← {0=unfilled,1=direction, 2=empty}
v96:DirDirectionValue ← string
v97:DirMeans ← {0=unfilled,1=destination, 2=path, 3=location, 4=empty}
v98:DirMeansValue ← string
v99:DirPrep ← {0=unfilled,1=to(the), 2=to(your), 3=empty}
v100:DirPrepValue ← string
v101:DirVerb ← {0=unfilled,1=go, 2=turn,3=bear,4=hang,5=move,6=empty}
v102:DirVerbValue ← string
v103:UserConfusions ← {0=none,1=one, 2=two or more}

dir_direction pre: v95==0
eff: v95 ← 1, v96 ← ‘direction’
dir_empty pre: v95==0
eff: v95 ← 2, v96 ← ‘empty’
means_destination pre: v97==0
eff: v97 ← 1, v98 ← ‘destination’
means_path pre: v97==0
eff: v97 ← 2, v98 ← ‘path’
means_location pre: v97==0
eff: v97 ← 3, v98 ← ‘location’
means_empty pre: v97==0
eff: v97 ← 4, v98 ← ‘empty’
prep_to(the) pre: v99==0
eff: v99 ← 1, v100 ← ‘to(the)’
prep_to(your) pre: v99==0
eff: v99 ← 2, v100 ← ‘to(your)’

```

$URules_2^3$

7. APPENDIX B

$URules_2^3$ (continued)	<pre> prep_empty pre: v99==0 eff: v99 ← 3, v100 ← ‘empty’ verb_go pre: v101==0 eff: v101 ← 1, v102 ← ‘go’ verb_turn pre: v101==0 eff: v101 ← 2, v102 ← ‘turn’ verb_bear pre: v101==0 eff: v101 ← 3, v102 ← ‘bear’ verb_hang pre: v101==0 eff: v101 ← 4, v102 ← ‘hang’ verb_move pre: v101==0 eff: v101 ← 5, v102 ← ‘move’ verb_empty pre: v101==0 eff: v101 ← 6, v102 ← ‘empty’ </pre>
--------------------------	--

Agent M_3^3 is responsible for the generation of orientation instructions. Similar to other surface realisation agents, it contains information on the open and filled decisions. It needs to choose a verb, a direction and an optional means constituent. The decisions that have been made are recorded in the information state.

IS_3^3	<pre> v104:Direction ← {0=unfilled,1=around, 2=round, 3=degrees, 4=empty} v105:DirectionValue ← <i>string</i> v106:OriMeans ← {0=unfilled,1=path, 2=destination, 3=empty} v107:OriMeansValue ← <i>string</i> v108:OriVerb ← {0=unfilled,1=turn, 2=you.want_to.turn} v109:OriVerbValue ← <i>string</i> v110:UserConfusions ← {0=none,1=one, 2=two or more} </pre>
----------	--

$URules_3^3$	<pre> dir_around pre: 104==0 eff: 104 ← 1, v105 ← ‘around’ dir_round pre: 104==0 eff: 104 ← 2, v105 ← ‘round’ dir_degrees pre: 104==0 eff: 104 ← 3, v105 ← ‘degrees’ dir_empty pre: 104==0 eff: 104 ← 4, v105 ← ‘empty’ means_path pre: 106==0 eff: 106 ← 1, v107 ← ‘path’ means_destination pre: 106==0 eff: 106 ← 2, v107 ← ‘destination’ means_empty pre: 106==0 eff: 106 ← 3, v107 ← ‘empty’ verb_turn pre: 108==0 eff: 108 ← 1, v109 ← ‘turn’ verb_you_want_to_turn pre: 108==0 eff: 108 ← 2, v109 ← ‘you_want_to_turn’ </pre>
--------------	---

Agent M_4^3 generates path instructions including a verb, a preposition, a relatum and an optional means of carrying out the instruction. The values of all decisions made are recorded in the information state. The update rules constrain

7. APPENDIX B

a decision point to being considered once per instruction.

IS_4^3	<pre> v111:PathMeans ← {0=unfilled,1=straight,2=destination,3=direction,4=empty} v112:PathMeansValue ← <i>string</i> v113:PathPrep ← {0=unfilled,1=through,2=along,3=across,4=empty, 5=down, 6=past} v114:PathPrepValue ← <i>string</i> v115:PathRelatum ← {0=unfilled,1=tunnel,2=space,3=room,4=landmark, 5=empty, 6=path} v116:PathRelatumValue ← <i>string</i> v117:PathVerb ← {0=unfilled,1=go,2=keep going,3=walk,4=pass, 5=empty, 6=continue} v118:PathVerbValue ← <i>string</i> v119:UserConfusions ← {0=none,1=one,2=two or more} means_straight pre: 111==0 eff: 111 ← 1, v112 ← ‘straight’ means_destination pre: 111==0 eff: 111 ← 2, v112 ← ‘destination’ means_direction pre: 111==0 eff: 111 ← 3, v112 ← ‘direction’ means_empty pre: 111==0 eff: 111 ← 4, v112 ← ‘empty’ prep_through pre: 113==0 eff: 113 ← 1, v114 ← ‘through’ prep_along pre: 113==0 eff: 113 ← 2, v114 ← ‘along’ prep_across pre: 113==0 eff: 113 ← 3, v114 ← ‘across’ </pre>
$URules_4^3$	

*URules*₄³(continued)

```
prep_empty pre: 113==0  
          eff: 113 ← 4, v114 ← ‘empty’  
prep_down pre: 113==0  
          eff: 113 ← 5, v114 ← ‘down’  
prep_past pre: 113==0  
          eff: 113 ← 6, v114 ← ‘past’  
verb_go pre: 117==0  
          eff: 117 ← 1, v118 ← ‘go’  
verb_keep-going pre: 117==0  
          eff: 117 ← 2, v118 ← ‘keep-going’  
verb_walk pre: 117==0  
          eff: 117 ← 3, v118 ← ‘walk’  
verb_pass pre: 117==0  
          eff: 117 ← 4, v118 ← ‘pass’  
verb_empty pre: 117==0  
          eff: 117 ← 5, v118 ← ‘empty’  
verb_continue pre: 117==0  
          eff: 117 ← 6, v118 ← ‘continue’  
relatum_tunnel pre: 115==0  
          eff: 115 ← 1, v116 ← ‘tunnel’  
relatum_space pre: 115==0  
          eff: 115 ← 2, v116 ← ‘space’  
relatum_room pre: 115==0  
          eff: 115 ← 3, v116 ← ‘room’  
relatum_landmark pre: 115==0  
          eff: 115 ← 4, v116 ← ‘landmark’
```

7. APPENDIX B

$URules_4^3(continued)$	<table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;">relatum_empty pre: 115==0</td><td></td></tr> <tr> <td>eff: 115 \leftarrow 5, v116 \leftarrow ‘empty’</td><td></td></tr> <tr> <td style="padding-right: 20px;">relatum_path pre: 115==0</td><td></td></tr> <tr> <td>eff: 115 \leftarrow 6, v116 \leftarrow ‘path’</td><td></td></tr> </table>	relatum_empty pre: 115==0		eff: 115 \leftarrow 5, v116 \leftarrow ‘empty’		relatum_path pre: 115==0		eff: 115 \leftarrow 6, v116 \leftarrow ‘path’	
relatum_empty pre: 115==0									
eff: 115 \leftarrow 5, v116 \leftarrow ‘empty’									
relatum_path pre: 115==0									
eff: 115 \leftarrow 6, v116 \leftarrow ‘path’									

Finally, agent M_5^3 is responsible for the generation of ‘straight’ instructions. It chooses a verb for the instruction, a way of realising the direction and optionally a means of executing the instruction. All decisions made are recorded in the information state.

IS_5^3	<table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;">v120:StrDirection $\leftarrow \{0=\text{unfilled}, 1=\text{straight}, 2=\text{forward},$</td><td></td></tr> <tr> <td>$3=\text{straight_ahead}, 4=\text{ahead}, 5=\text{empty}\}$</td><td></td></tr> <tr> <td>v121:StrDirectionValue $\leftarrow string$</td><td></td></tr> <tr> <td style="padding-right: 20px;">v122:StrMeans $\leftarrow \{0=\text{unfilled}, 1=\text{direction}, 2=\text{destination},$</td><td></td></tr> <tr> <td>$3=\text{orientation}, 4=\text{empty}\}$</td><td></td></tr> <tr> <td>v123:StrMeansValue $\leftarrow string$</td><td></td></tr> <tr> <td style="padding-right: 20px;">v124:StrVerb $\leftarrow \{0=\text{unfilled}, 1=\text{go}, 2=\text{empty}, 3=\text{keep_going},$</td><td></td></tr> <tr> <td>$4=\text{you_want_to_go}\}$</td><td></td></tr> <tr> <td>v125:StrVerbValue $\leftarrow string$</td><td></td></tr> <tr> <td>v126:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$</td><td></td></tr> </table>	v120:StrDirection $\leftarrow \{0=\text{unfilled}, 1=\text{straight}, 2=\text{forward},$		$3=\text{straight_ahead}, 4=\text{ahead}, 5=\text{empty}\}$		v121:StrDirectionValue $\leftarrow string$		v122:StrMeans $\leftarrow \{0=\text{unfilled}, 1=\text{direction}, 2=\text{destination},$		$3=\text{orientation}, 4=\text{empty}\}$		v123:StrMeansValue $\leftarrow string$		v124:StrVerb $\leftarrow \{0=\text{unfilled}, 1=\text{go}, 2=\text{empty}, 3=\text{keep_going},$		$4=\text{you_want_to_go}\}$		v125:StrVerbValue $\leftarrow string$		v126:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$	
v120:StrDirection $\leftarrow \{0=\text{unfilled}, 1=\text{straight}, 2=\text{forward},$																					
$3=\text{straight_ahead}, 4=\text{ahead}, 5=\text{empty}\}$																					
v121:StrDirectionValue $\leftarrow string$																					
v122:StrMeans $\leftarrow \{0=\text{unfilled}, 1=\text{direction}, 2=\text{destination},$																					
$3=\text{orientation}, 4=\text{empty}\}$																					
v123:StrMeansValue $\leftarrow string$																					
v124:StrVerb $\leftarrow \{0=\text{unfilled}, 1=\text{go}, 2=\text{empty}, 3=\text{keep_going},$																					
$4=\text{you_want_to_go}\}$																					
v125:StrVerbValue $\leftarrow string$																					
v126:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$																					
$URules_5^3$	<table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;">dir_straight pre: v120==0</td> <td></td> </tr> <tr> <td>eff: v120 \leftarrow 1, v121 \leftarrow ‘straight’</td> <td></td> </tr> <tr> <td style="padding-right: 20px;">dir_forward pre: v120==0</td> <td></td> </tr> <tr> <td>eff: v120 \leftarrow 2, v121 \leftarrow ‘forward’</td> <td></td> </tr> <tr> <td style="padding-right: 20px;">dir_straight_ahead pre: v120==0</td> <td></td> </tr> <tr> <td>eff: v120 \leftarrow 3, v121 \leftarrow ‘straight_ahead’</td> <td></td> </tr> </table>	dir_straight pre: v120==0		eff: v120 \leftarrow 1, v121 \leftarrow ‘straight’		dir_forward pre: v120==0		eff: v120 \leftarrow 2, v121 \leftarrow ‘forward’		dir_straight_ahead pre: v120==0		eff: v120 \leftarrow 3, v121 \leftarrow ‘straight_ahead’									
dir_straight pre: v120==0																					
eff: v120 \leftarrow 1, v121 \leftarrow ‘straight’																					
dir_forward pre: v120==0																					
eff: v120 \leftarrow 2, v121 \leftarrow ‘forward’																					
dir_straight_ahead pre: v120==0																					
eff: v120 \leftarrow 3, v121 \leftarrow ‘straight_ahead’																					

*URules*₅³(continued)

<code>dir_ahead pre: v120==0 eff: v120 ← 4, v121 ← ‘ahead’</code> <code>dir_empty pre: v120==0 eff: v120 ← 5, v121 ← ‘empty’</code> <code>means_direction pre: v122==0 eff: v122 ← 1, v123 ← ‘direction’</code> <code>means_destination pre: v122==0 eff: v122 ← 2, v123 ← ‘destination’</code> <code>means_orientation pre: v122==0 eff: v122 ← 3, v123 ← ‘orientation’</code> <code>means_empty pre: v122==0 eff: v122 ← 4, v123 ← ‘empty’</code> <code>verb_go pre: v124==0 eff: v124 ← 1, v125 ← ‘go’</code> <code>verb_keep_going pre: v124==0 eff: v124 ← 2, v125 ← ‘keep-going’</code> <code>verb_you_want_to_go pre: v124==0 eff: v124 ← 3, v125 ← ‘you_want_to_go’</code>
--

The specified information state can be used to constraint the agent’s learning process in a way so as to reflect strong human preferences for the target domain. Note however that not all human preferences identified in Sections 4.2.1.1 and 4.2.1.2 have been implemented in the update rules of the hierarchical information state. In terms of navigation, it was left to the learning agent to choose a navigation level (among low, high and mixed), and in terms of referring expressions, we only implemented the preference for colour that humans showed. The full set of rules learnt using decision trees was used as a baseline to compare the fully-learnt behaviour of our learning agent and the semi-learnt behaviour resulting from the

7. APPENDIX B

combination of hierarchical RL and the hierarchical IS.

Appendix C

In this section we will revise our hierarchy of learning agents to include extended models for surface realisation. Agents $M_{0\dots 5}^3$ will be adapted to include the notion of a *constituent alignment score* (*CAS*) so that they can be optimised in accordance with it. To this end, we will add three state variables to each surface realisation agent: one variable indicating the last surface variant chosen, one variable indicating the *CAS* for this surface variant and one variable indicating the *CAS* for all other surface variants (i.e. that were not chosen in the last instruction). This allows the learning agent to be aware of the proportion of alignment and variation and take it into account for its subsequent decision making. We distinguish four values that the *CAS* can take on: *unaligned*, *low*, *medium* and *high*, where *medium* is the value that will receive the highest rewards. It lies between 0.4 and 0.6. A value of *unaligned* corresponds to a *CAS* of 0, a *low CAS* is below 0.4 and a *high CAS* is above 0.6. Importantly, we will make use of the fact that surface form variants occur in sequences of different likelihoods in the human corpus. It is therefore not necessary to introduce a *CAS* for every semantic concept (and its associated realisation variants), i.e. one for ‘verb’, one for ‘direction’, etc. Instead, we can introduce just one single *CAS* for the verb of an instruction and leave the remaining optimisation of the word sequence to probabilities obtained during learning. The objective is that the agent will then automatically learn those surface forms that are most likely to co-occur with the chosen verb. The details of how the graphical models are integrated into the learning framework by providing feedback during learning was addressed in Section 5.4.1. The information state will not be extended since we will leave the

7. APPENDIX C

optimisation of surface realisation exclusively to the learning agent.

S_0^3	v1:CASReVerb $\leftarrow \{0=\text{unaligned}, 1=\text{low}, 2=\text{medium}, 3=\text{high}\}$ v2:CASTherOtherReVerb $\leftarrow \{0=\text{unaligned}, 1=\text{low}, 2=\text{medium}, 3=\text{high}\}$ v3:Distractor $\leftarrow \{0=\text{true}, 1=\text{false}\}$ v4:Landmark $\leftarrow \{0=\text{true}, 1=\text{false}\}$ v5:Position $\leftarrow \{0=\text{true}, 1=\text{false}\}$ v6:PreviousReVerb $\leftarrow \{0=\text{unfilled}, 1=\text{push}, 2=\text{press}, 3=\text{click}, 4=\text{click_on}, 5=\text{choose}, 6=\text{get}, 7=\text{hit}, 8=\text{empty}\}$ v7:ReDeterminer $\leftarrow \{0=\text{unfilled}, 1=\text{the}, 2=\text{that}, 3=\text{empty}\}$ v8:ReSpatialRelation $\leftarrow \{0=\text{unfilled}, 1=\text{adv}, 2=\text{pp}, 3=\text{rel_clause_pp}\}$ v9:ReType $\leftarrow \{0=\text{unfilled}, 1=\text{button}, 2=\text{one}, 3=\text{it}, 4=\text{empty}\}$ v10:ReVerb $\leftarrow \{0=\text{unfilled}, 1=\text{push}, 2=\text{press}, 3=\text{click}, 4=\text{click_on}, 5=\text{choose}, 6=\text{get}, 7=\text{hit}, 8=\text{empty}\}$ v11:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$
A_0^3	det_the, det_that, det_empty, sr_adv, sr_pp, sr_rel_clause_pp, type_button, type_one, type_empty, type_it, verb_push, verb_press, verb_click, verb_click_on, verb_choose, verb_get, verb_hit, verb_empty
G_0^3	$\left[\begin{array}{l} v1=2, v2=2, v7=0, v8=0, v9=0, v10=0 \end{array} \right]$

As before, agent M_0^3 is responsible for the surface realisation of referring expressions. The state representation has not changed except for the additional features ‘CASReVerb’, ‘CASTherOtherReVerb’ and ‘PreviousReVerb’. The first of these features indicates the *CAS* of the last verb that the agent chose. The second feature correspondingly indicates the *CAS* of the set of all other verbs that were chosen in the last instruction. The third variable holds the value of the last

verb so that the agent could use this information to choose the next verb; i.e. based on the previous verb choice and the *CAS* that the choice yielded. In this way, the agent can learn to balance the trade-offs of variation and alignment and generate natural and human-like surface forms.

S_1^3	<pre> v12:CasDesVerb ← {0=unaligned,1=low,2=medium,3=high} v13:CasOtherDesVerb ← {0=unaligned,1=low,2=medium,3=high} v14:DesDirection ← {0=unfilled,1=direction, 2=straight, 3=empty} v15:DesPrep ← {0=unfilled,1=to, 2=towards, 3=empty, 4=into, 5=in, 6=until} v16:DesRelatum ← {0=unfilled,1=room, 2=landmark, 3=empty} v17:DesVerb ← {0=unfilled,1=go, 2=keep-going, 3=empty, 4=get, 5=return, 6=continue, 7=walk} v18:PreviousDesVerb ← {0=unfilled,1=go, 2=keep-going, 3=empty, 4=get, 5=return, 6=continue, 7=walk} v19:UserConfusions ← {0=none,1=one, 2=two or more} </pre>
A_1^3	<pre> dir_direction, dir_straight, dir_empty, prep_to, prep_towards, prep_empty, prep_into, prep_in, prep_until, relatum_room, relatum_landmark, relatum_empty, verb_continue, verb_go, verb_walk, verb_get, verb_return, verb_empty, verb_keep-going </pre>
G_1^3	$\left[\begin{array}{l} v12=2, v13=2, v14=0, v15=0, v16=0, v17=0 \end{array} \right]$

The remaining trade-offs of user satisfaction, task success and cognitive load are still subject to optimisation as well since no other state variables were removed or added. The information state was not augmented with the new state variables and therefore still just ensures that the agent does not consider any decision point more than once. It thus prevents it from entering into loops. The goal state of the referring expression generation agent is reached when the agent has chosen surface forms for the verb of the referring expression, the determiner and the

7. APPENDIX C

type.

S_2^3	v19:CASDirVerb $\leftarrow \{0=\text{unaligned}, 1=\text{low}, 2=\text{medium}, 3=\text{high}\}$ v20:CA SOtherDirVerb $\leftarrow \{0=\text{unaligned}, 1=\text{low}, 2=\text{medium}, 3=\text{high}\}$ v21:DirDirection $\leftarrow \{0=\text{unfilled}, 1=\text{direction}, 2=\text{empty}\}$ v22:DirMeans $\leftarrow \{0=\text{unfilled}, 1=\text{destination}, 2=\text{path}, 3=\text{location}, 4=\text{empty}\}$ v23:DirPrep $\leftarrow \{0=\text{unfilled}, 1=\text{to(the)}, 2=\text{to(your)}, 3=\text{empty}\}$ v24:DirVerb $\leftarrow \{0=\text{unfilled}, 1=\text{go}, 2=\text{turn}, 3=\text{bear}, 4=\text{hang}, 5=\text{move}, 6=\text{empty}\}$ v25:PreviousDirVerb $\leftarrow \{0=\text{unfilled}, 1=\text{go}, 2=\text{turn}, 3=\text{bear}, 4=\text{hang}, \}$ $\quad \quad \quad \{5=\text{move}, 6=\text{empty}\}$ v26:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$
A_2^3	dir_direction, dir_empty, means_destination, means_path, means_location, means_empty, prep_to(the), prep_to(your), prep_empty, verb_go, verb_turn, verb_bear, verb_move, verb_hang, verb_empty
G_2^3	$\left[\begin{array}{l} v19=2, v20=2, v21=0, v22=0, v23=0, v24=0 \end{array} \right]$

S_3^3	v27:CA SOriVerb $\leftarrow \{0=\text{unaligned}, 1=\text{low}, 2=\text{medium}, 3=\text{high}\}$ v28:CA SOtherOriVerb $\leftarrow \{0=\text{unaligned}, 1=\text{low}, 2=\text{medium}, 3=\text{high}\}$ v29:Direction $\leftarrow \{0=\text{unfilled}, 1=\text{around}, 2=\text{round}, 3=\text{degrees}, 4=\text{empty}\}$ v30:OriMeans $\leftarrow \{0=\text{unfilled}, 1=\text{path}, 2=\text{destination}, 3=\text{empty}\}$ v31:OriVerb $\leftarrow \{0=\text{unfilled}, 1=\text{turn}, 2=\text{you_want_to_turn}\}$ v32:PreviousOriVerb $\leftarrow \{0=\text{unfilled}, 1=\text{turn}, 2=\text{you_want_to_turn}\}$ v33:UserConfusions $\leftarrow \{0=\text{none}, 1=\text{one}, 2=\text{two or more}\}$
A_3^3	dir_around, dir_round, dir_degrees, dir_empty, means_path, means_empty, means_destination, verb_turn, verb_you_want_to_turn
G_3^3	$\left[\begin{array}{l} v27=2, v28=2, v29=0, v30=0, v31=0 \end{array} \right]$

Agent M_1^3 is responsible for the surface realisation of destination instructions.

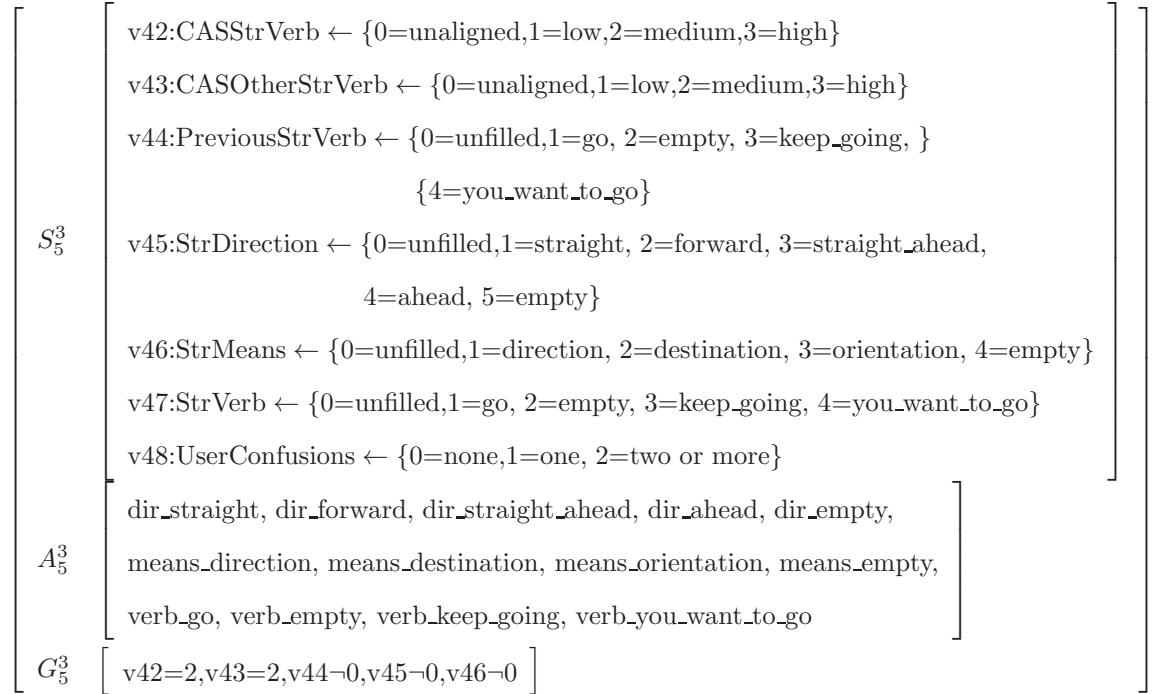
It also has a notion of the *CAS* of the last verb chosen as well as of all verbs chosen so far. In addition it has a variable for the surface form of the previous verb. It reaches its goal state when a verb, a direction, a preposition and a relatum have been chosen.

S_4^3	<pre> v34:CASEPathVerb ← {0=unaligned,1=low,2=medium,3=high} v35:CASEOtherPathVerb ← {0=unaligned,1=low,2=medium,3=high} v36:PathMeans ← {0=unfilled,1=straight,2=destination,3=direction,4=empty} v37:PathPrep ← {0=unfilled,1=through,2=along,3=across,4=empty, 5=down, 6=past} v38:PathRelatum ← {0=unfilled,1=tunnel,2=space,3=room,4=landmark, 5=empty, 6=path} v39:PathVerb ← {0=unfilled,1=go,2=keep-going,3=walk,4=pass, 5=empty, 6=continue} v40:PreviousPathVerb ← {0=unfilled,1=go,2=keep-going,3=walk,4=pass, 5=empty, 6=continue} v41:UserConfusions ← {0=none,1=one, 2=two or more} means_straight, means_empty, means_destination, means_direction, prep_empty prep_through, prep_along, prep_across, prep_empty, prep_down, prep_past, verb_go, verb_keep_going, verb_walk, verb_pass, verb_empty, verb_continue, relatum_space, relatum_landmark, relatum_tunnel, relatum_room, relatum_path, relatum_empty </pre>
A_4^3	
G_4^3	$\left[v34=2, v35=2, v36=0, v37=0, v38=0, v39=0 \right]$

Agent M_2^3 generates the surface forms for direction instructions. It shares the properties of the previous two agents in that it optimises surface realisation with respect to a *CAS*. It reaches its goal state when a verb, a preposition, a direction

7. APPENDIX C

and a means of the instruction have been chosen.



Agent M_4^3 realises surface forms of orientation instructions. It optimises them with respect to a *CAS* and reaches its goal state when a verb, a direction and a means have been realised. Path instructions are generated by agent M_4^3 . It keeps the value of the verb chosen in the previous instruction as well as its *CAS* and the *CAS* of the set of all remaining verbs. Based on this, it optimises the proportion of variation and alignment of its output. The agent reaches its goal state when a verb, a preposition, a means and a relatum have been chosen. Agent M_5^3 finally is responsible for the surface realisation of instructions of type ‘straight’. It optimises them with respect to a *CAS* and reaches its goal state when a verb, a direction and a means of the instruction have been chosen.

References

- Gabor Angeli, Percy Liang, and Dan Klein. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, MIT Stata Center, Massachusetts, USA, 2010. [4](#), [27](#), [29](#), [191](#)
- Douglas Appelt. *Planning English Sentences*. Cambridge University Press, New York, NY, USA, 1992. [15](#)
- Srinivas Bangalore and Owen Rambow. Exploiting a Probabilistic Hierarchical Model for Generation. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 42–48, 2000a. [17](#)
- Srinivas Bangalore and Owen Rambow. Corpus-Based Lexical Choice in Natural Language Generation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 464–471, Hong Kong, 2000b. [17](#)
- Andrew G. Barto and Sridhar Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, 13, 2003. [50](#), [52](#), [56](#)
- Regina Barzilay and Lillian Lee. Bootstrapping Lexical Choice via Multiple-Sequence Alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Pennsylvania, Philadelphia, USA, 2002. [30](#), [105](#)

REFERENCES

- John Bateman. Enabling Technology for Multilingual Natural Language Generation: the KPML (d)evelopment Environment. *Natural Language Engineering*, 3(1):15–55, 1997. [13](#)
- John Bateman and Cécile Paris. Phrasing a Text in Terms the User Can Understand. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1511–1517, Detroit, Michigan, USA, 1989. [13](#)
- John Bateman and Elke Teich. Selective Information Presentation in an Integrated Publication System: An Application of Genre-Driven Text Generation. *Information Processing and Management: An International Journal*, 31(5):753–767, September 1995. [13](#)
- Anja Belz. Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models. *Natural Language Engineering*, 14(4):431–455, 2008. [31](#), [105](#)
- Anja Belz and Ehud Reiter. Comparing Automatic and Human Evaluations of NLG Systems. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, 2006. [83](#), [105](#), [109](#)
- Luciana Benotti and Alexandre Denis. CL system: Giving instructions by corpus based selection. In *Proceedings of the Generation Challenges Sessions at the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011a. [165](#), [167](#)
- Luciana Benotti and Alexandre Denis. Giving Instructions in Virtual Environments by Corpus-Based Selection. In *Proceedings of the 12th Annual Meeting*

REFERENCES

on Discourse and Dialogue (SIGdial), Portland, Oregon, 2011b. [xxxi](#), [165](#), [166](#), [183](#)

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. [30](#), [118](#), [121](#), [123](#)

Kathryn Bock. Syntactic persistence in language production. *Cognitive Psychology*, 18:355–387, 1986. [108](#)

Peter Bohlin, Robin Cooper, Elisabet Engdahl, and Staffan Larsson. Information States and Dialogue Moves Engines. In *IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Stockholm, Sweden, 1999. [91](#)

Kalina Bontcheva and Yorick Wilks. Dealing with Dependencies between Content Planning and Surface Realisation in a Pipeline Generation Architecture. In *In Proceedings of International Joint Conference in Artificial Intelligence (IJCAI'01*, pages 7–10, 2001. [4](#), [29](#)

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information State Update Dialogue System Architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, 2003. [91](#), [97](#)

Thomas Bouttaz, Edoardo Pignotti, Chris Mellish, and Peter Edwards. A Policy-Based Approach to Context Dependent Natural Language Generation. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [20](#)

REFERENCES

- S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement Learning for Mapping Instructions to Actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore, 2009. [24](#)
- Holly Branigan, Martin Pickering, and Alexandra Cleland. Syntactic coordination in dialogue. *Cognition*, 75:B13–25, 2000. [108](#)
- Johannes Braunias, Uwe Boltz, Markus Draeger, Boris Fersing, and Olga Nikitina. The GIVE-2 Challenge: Saarland NLG System. In *Proceedings of the Generation Challenges Sessions at the 6th International Conference on Natural Language Generation (INLG)*, Dublin, Ireland, 2010. [164](#)
- Susan Brennan and Herbert Clark. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22:1482–1493, 1996. [107](#)
- Ivan Bulyko and Mari Ostendorf. Efficient integrated response generation from multiple targets using weighted finite state transducers. *Computer Speech and Language*, 16:533–550, 2002. [26](#), [27](#)
- Okko Buss and David Schlangen. Modelling Sub-Utterance Phenomena in Spoken Dialogue Systems. In *Proceedings the Workshop on the Semantics and Pragmatics of Dialogue (SemDIAL / PozDial)*, Poznan, Poland, 2010. [192](#)
- Okko Buss and David Schlangen. DIUM—An Incremental Dialogue Manager That Can Produce Self-Corrections. In *Proceedings the Workshop on the Se-*

REFERENCES

mantics and Pragmatics of Dialogue (SemDIAL / Los Angelogue), Los Angeles, CA, USA, 2011. [192](#)

Donna Byron, Alexander Koller, Kristina Striegnitz, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. Report on the First NLG Challenge on Generating Instructions in Virtual Environments (GIVE). In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG)*, Athens, Greece, 2009. [xxi, 7, 59](#)

Jonathan Calder, Mike Reape, and Henk Zeevat. An algorithm for generation in Unification Categorial Grammar. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Manchester, UK, 1989. [13](#)

Jean Carletta. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22(2):249–354, 1996. [63](#)

Eugene Charniak. Bayesian Networks Without Tears. *AI MAGAZINE*, 12(4): 50–63, 1991. [xxiv, 118, 119, 121](#)

David L. Chen, Joohyun Kim, and Raymond J. Mooney. Training a Multilingual Sportscaster: Using Perceptual Context to Learn Language. *Journal of Artificial Intelligence Research*, 37:397–435, 2010. [19](#)

Christian Chiarcos. Evaluating Salience Metrics for the Context-Adequate Realization of Discourse Referents. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [20](#)

Noam Chomsky. *Syntactic Structure*. London: Mouton, 1957. [112](#)

REFERENCES

- Herbert Clark and Deanna Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986. [107](#)
- Philip Cohen and C. Raymond Perrault. *Elements of a Plan-Based Theory of Speech Acts*, pages 423–440. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986. [15](#)
- William Cohen. Learning Trees and Rules with Set-valued Features. In *Proceedings of the 14th Conference of the American Association for Artificial Intelligence (AAAI)*, Portland, Oregon, USA, 1996. [23](#)
- Fabio Cozman. Generalizing variable elimination in Bayesian networks. In *IBERAMIA/SBIA, Workshop on Probabilistic Reasoning in Artificial Intelligence*, Sao Paulo, Brazil, 2000. [121](#)
- Heriberto Cuayáhuitl. *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. PhD Thesis, University of Edinburgh, Scotland, School of Informatics, 2009. [xxi](#), [52](#), [56](#), [84](#)
- Heriberto Cuayáhuitl. Learning Dialogue Agents with Bayesian Relational State Representations. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI-KRDPS)*, Barcelona, Spain, 2011. [140](#), [171](#)
- Heriberto Cuayáhuitl and Nina Dethlefs. Spatially-aware Dialogue Control Using Hierarchical Reinforcement Learning. *ACM Transactions on Speech and Language Processing (Special Issue on Machine Learning for Robust and Adaptive Spoken Dialogue System)*, 7(3), 2011a. [xxvii](#), [xxviii](#), [28](#), [29](#), [100](#), [140](#), [171](#), [176](#), [177](#), [178](#), [179](#), [180](#), [181](#), [191](#), [192](#)

REFERENCES

- Heriberto Cuayáhuitl and Nina Dethlefs. Optimizing Situated Dialogue Management in Unknown Environments. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Florence, Italy, 2011b. [68](#), [100](#), [140](#), [171](#), [192](#)
- Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. Human-Computer Dialogue Simulation Using Hidden Markov Models. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU)*, San Juan, Puerto Rico, 2005. [134](#)
- Heriberto Cuayáhuitl, Nina Dethlefs, Kai-Florian Richter, Thora Tenbrink, and John Bateman. A Dialogue System for Indoor Wayfinding Using Text-Based Natural Language. *International Journal of Computational Linguistics and Applications*, 1:285–304, 2010. [68](#)
- Robert Dale. Cooking Up Referring Expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, British Columbia, Canada, 1989. [12](#)
- Robert Dale and Nicholas Haddock. Generating Referring Expressions Involving Relations. In *Proceedings of the 5th Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*, Berlin, Germany, 1991. [12](#)
- Robert Dale and Ehud Reiter. Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science*, 19(2): 233–263, 1995. [12](#)
- Robert Dale and Jette Viethen. Referring Expression Generation Through

REFERENCES

- Attribute-Based Heuristics. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG)*, Athens, Greece, 2009. [12](#), [83](#), [86](#)
- Vera Demberg and Johanna Moore. Information Presentation in Spoken Dialogue Systems. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 353–360, Trento, Italy, 2006. [18](#)
- Alexandre Denis. Generating Referring Expressions with Reference Domain Theory. In *Proceedings of the 6th International Natural Language Generation Conference (INLG)*, Dublin, Ireland, 2010. [164](#)
- Alexandre Denis. The Loria instruction generation system L in GIVE-2.5. In *Proceedings of the Generation Challenges Sessions at the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [164](#), [167](#)
- Nina Dethlefs. The Bremen System for the GIVE-2.5 Challenge. In *Proceedings of the Generation Challenges Sessions at the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [181](#)
- Nina Dethlefs and Heriberto Cuayahuitl. Hierarchical Reinforcement Learning for Adaptive Text Generation. In *Proceedings of the 6th International Natural Language Generation Conference (INLG)*, Dublin, Ireland, 2010. [111](#)
- Nina Dethlefs and Heriberto Cuayahuitl. Hierarchical Reinforcement Learning and Hidden Markov Models for Task-Oriented Natural Language Generation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, Portland, Oregon, USA, 2011a. [105](#), [191](#)

REFERENCES

- Nina Dethlefs and Heriberto Cuayáhuitl. Combining Hierarchical Reinforcement Learning and Bayesian Networks for Natural Language Generation in Situated Dialogue. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011b. [105](#), [191](#)
- Nina Dethlefs and Heriberto Cuayáhuitl. Comparing HMMs and Bayesian Networks for Surface Realisation. In *Proceedings of the 12th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Montréal, Canada, 2012. [138](#)
- Nina Dethlefs, Heriberto Cuayáhuitl, Kai-Florian Richter, Elena Andonova, and John Bateman. Evaluating Task Success in a Dialogue System for Indoor Navigation. In *Proceedings of the 14th Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*, Poznan, Poland, 2010. [69](#), [70](#)
- Nina Dethlefs, Yunhui Wu, Aisan Kazerani, and Stephan Winter. Generation of Adaptive Route Descriptions in Urban Environments. *Spatial Cognition and Computation*, 11(2):153–177, 2011. [14](#)
- Nina Dethlefs, Helen Hastie, Heriberto Cuayáhuitl, and Oliver Lemon. Conditional Random Fields for Responsive Surface Realisation Using Global Features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria, 2013. [32](#)
- Thomas G. Dietterich. An Overview of MAXQ Hierarchical Reinforcement Learning. In *Symposium on Abstraction, Reformulation, and Approximation (SARA)*, HorseShoeBay, Texas, USA, 2000a. [54](#)

REFERENCES

- Thomas G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13, 2000b. [52](#), [54](#), [56](#)
- Thomas G. Dietterich. State Abstraction in MAXQ Hierarchical Reinforcement Learning. In S.A. Solla, T.K. Leen, and K.-R. Mueller, editors, *Advances in Neural Information Processing Systems*, pages 994–1000. MIT Press, Cambridge, Massachusetts, 2000c. [41](#), [52](#)
- Myroslava Dzikovska, Johanna Moore, Natalie Steinauser, and Gwendolyn Campbell. Exploring User Satisfaction in a Tutorial Dialogue System. In *Proceedings of the 12th Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2001. [160](#)
- Michael Elhadad. *Using Argumentation to Control Lexical Choice: a Functional Unification-based approach*. PhD thesis, Department of Computer Science, Columbia University, New York, 1993. [13](#)
- Mary Ellen Foster and Jon Oberlander. Data-Driven Generation of Emphatic Facial Displays. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 353–360, Trento, Italy, 2006. [17](#), [106](#), [109](#)
- Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. The GIVE-2 Corpus of Generating Instructions in Virtual Environments. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Malta, 2010. [60](#)

REFERENCES

- Konstantina Garoufi and Alexander Koller. Automated Planning for Situated Natural Language Generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, 2010. [16](#), [83](#), [165](#)
- Konstantina Garoufi and Alexander Koller. Combining symbolic and corpus-based approaches for the generation of successful referring expressions. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011a. [16](#), [83](#), [183](#)
- Konstantina Garoufi and Alexander Koller. The Potsdam NLG systems at the GIVE-2.5 Challenge. In *In Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011b. [164](#), [167](#), [183](#)
- Simon Garrod and Anthony Anderson. Saying What You Mean in Dialogue: A Study in conceptual and semantic co-ordination. *Cognition*, 27(2):181–218, 1987. [107](#)
- Simon Garrod and Aileen Clark. The development of dialogue co-ordination skills in schoolchildren. *Language and Cognitive Processes*, 8(1):101–126, 1993. [107](#)
- Simon Garrod and Gwyneth Doherty. Conversation, co-ordination and convention: An empirical investigation of how groups establish linguistic conventions. *Cognition*, 53(3):181–215, 1994. [107](#)
- Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. On-line policy optimisation of spoken dialogue systems via interaction with human sub-

REFERENCES

- jects. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU)*, Waikoloa, Hawaii, USA, 2011. [68](#), [192](#)
- Kallirroi Georgila, Nikos Fakotakis, and George Kokkinakis. Stochastic Language Modelling for Recognition and Generation in Dialogue Systems. *TAL (Traitement automatique des langues) Journal*, 43(3):129–154, 2002. [31](#), [105](#)
- Zoubin Ghahramani. Learning Dynamic Bayesian Networks. *Lecture Notes in Computer Science*, 1387:168–197, 1997. [123](#)
- Zoubin Ghahramani. An Introduction to Hidden Markov Models and Bayesian Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42, 2001. [120](#), [123](#)
- Dave Golland, Percy Liang, and Dan Klein. A Game-Theoretic Approach to Generating Spatial Descriptions. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, MIT, Massachusetts, USA, 2010. [20](#)
- Elias Gyftodimos and Peter Flach. Hierarchical Bayesian Networks: A Probabilistic Reasoning Model for Structured Domains. In Edwin de Jong and Tim Oates, editors, *Proceedings of the ICML-2002 Workshop on Development of Representations*, pages 23–30. ISSN 0733419348, 2002. [123](#)
- Ruqaiya Hasan. Meaning, context and text: fifty years after Malinowski. In James D. Benson and William S. Greaves, editors, *Systemic Perspectives on Discourse, Volume 1*. Ablex, Norwood, New Jersey, 1985. [2](#)

REFERENCES

- Sarah Haywood, Martin Pickering, and Holly Branigan. Do Speakers Avoid Ambiguity During Dialogue? *Psychological Science*, 16:362–366, 2005. [108](#)
- Peter Heeman. Combining Reinforcement Learning with Information-State Update Rules. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 268–275, Rochester, NY, USA, 2007. [84](#), [103](#)
- Peter Heeman and Graeme Hirst. Collaborating on Referring Expressions. *Computational Linguistics*, 21:351–382, September 1995. [15](#)
- James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets. *Computational Linguistics*, 34(4):487–511, 2008. [21](#)
- Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christine Fellbaum, editor, *WordNet: An Electronic Database and Some of its Applications*, pages 305–332. Cambridge, MA: MIT Press, 1998. [111](#)
- Kate Hone and Robert Graham. Towards a Tool for the Subjective Assessment of Speech System Interfaces (SASSI). *Natural Language Engineering*, 6(3-4): 287–303, 2000. [160](#)
- Helmut Horacek. Generating Referential Descriptions Under Conditions of Uncertainty. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG)*, Aberdeen, Scotland, 2005. [12](#)
- Eduard Hovy. Approaches to the Planning of Coherent Text. In Cecile Paris,

REFERENCES

- Willian Swartout, and William Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 83–102. Kluwer Academic Publishers, Boston, Massachusetts, 1991. [15](#)
- Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Upper Saddle River, NJ, USA, Prentice Hall PTR, 2001. [123](#)
- Tommi Jaakkola, Michael Jordan, and Satinder Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6), 1994. [46](#)
- Ray Jackendoff. *Foundations of Language*. Oxford University Press, 2002. [107](#)
- Manfred Jaeger. Relational Bayesian Networks. In *Proceedings of 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 266–273, Providence, Rhode Island, USA, 1997. [123](#)
- Srinivasan Janarthanam. *Learning User Modelling Strategies for Adaptive Referring Expression Generation in Spoken Dialogue Systems*. PhD thesis, University of Edinburgh, School of Informatics, Scotland, 2011. [21](#)
- Srinivasan Janarthanam and Oliver Lemon. A Wizard-of-Oz environment to study Referring Expression Generation in a Situated Spoken Dialogue Task. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Athens, Greece, 2009a. [22](#)
- Srinivasan Janarthanam and Oliver Lemon. A Two-tier User Simulation Model for Reinforcement Learning of Adaptive Referring Expression Generation Poli-

REFERENCES

cies. In *Proceedings of the 10th Annual Meeting on Discourse and Dialogue (SIGdial)*, Queen Mary University, London, UK, 2009b. [22](#)

Srinivasan Janarthanam and Oliver Lemon. Learning to Adapt to Unknown Users: Referring Expression Generation in Spoken Dialogue Systems. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics (ACL)*, Uppsala, Sweden, 2010a. [21](#)

Srinivasan Janarthanam and Oliver Lemon. Adaptive Referring Expression Generation in Spoken Dialogue Systems: Evaluation with Real Users. In *Proceedings of the 11th Annual Meeting on Discourse and Dialogue (SIGdial)*, Tokyo, Japan, 2010b. [22](#)

Srinivasan Janarthanam, Helen Hastie, Oliver Lemon, and Xingkun Liu. “The day after the day after tomorrow?” A machine learning approach to adaptive temporal expression generation: training and evaluation with real users. In *Proceedings of the 12th Annual Meeting on Discourse and Dialogue (SIGdial)*, Portland, Oregon, USA, 2011. [23](#)

Finn Jensen. *An Introduction to Bayesian Networks*. Springer Verlag, New York, 1996. [121](#)

Aravind Joshi. The Relevance of Tree Adjoining Grammar to Natural Language Generation. In G Kempen, editor, *Natural Language Generation*, Dordrecht, The Netherlands, 1987. Martinus Nijhoff Peess. [13](#)

Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computation*

REFERENCES

- tional Linguistics*. 2nd Edition. Prentice-Hall, 2009. xxiv, 25, 114, 115, 116, 118, 134
- Leslie Pack Kaelbling, Michael Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 41
- Martin Kay. Unification in Grammar. In V Dahl and P Saint-Dizier, editors, *Natural Language Understanding and Logic Programming*, Amsterdam, 1985. North-Holland. 13
- Alexander Koller and Ronald Petrick. Experiences with Planning for Natural Language Generation. *Computational Intelligence*, 27(1):23–40, 2011. 16
- Alexander Koller and Matthew Stone. Sentence generation as planning. In *Proceedings of 47th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, 2007. 16
- Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, and Johanna Moore. The First Challenge on Generating Instructions in Virtual Environments. In M. Theune and E. Krahmer, editors, *Empirical Methods in Natural Language Generation*, pages 337–361. Springer Verlag, Berlin/Heidelberg, 2010. xxvi, 59, 162, 164
- Daphne Koller and Avi Pfeffer. Object-Oriented Bayesian Networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 302–313, Providence, Rhode Island, USA, 1997. 123

REFERENCES

- Ioannis Konstas and Mirella Lapata. Concept-to-Text Generation via Discriminative Reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea, 2012. [32](#)
- Solomon Kullback and Richard Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951. [134](#)
- Irene Langkilde and Kevin Knight. Generation that Exploits Corpus-Based Statistical Knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 704–710, 1998. [17](#)
- Staffan Larsson and David Traum. Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3–4):323–340, 2000. [91](#)
- Benoit Lavoie and Owen Rambow. A Fast and Portable Realizer for Text Generation Systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C, 1997. [13](#)
- Oliver Lemon. Adaptive Natural Language Generation in Dialogue using Reinforcement Learning. In *Proceedings the Workshop on the Semantics and Pragmatics of Dialogue (SemDIAL / Londial)*, London, UK, 2008. [21](#)
- Oliver Lemon. Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation. *Computer Speech and Language*, 25(2), 2011. [27](#), [29](#), [191](#)

REFERENCES

- Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. An ISU Dialogue System Exhibiting Reinforcement Learning of Dialogue Policies: Generic Slot-filling in the TALK In-car System. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, 2006. [91](#)
- Willem Levelt. *Speaking: From Intenion to Articulation*. MIT Press, 1989. [109](#), [111](#)
- Willem Levelt and Stephanie Kelter. Surface form and memory in question answering. *Cognitive Psychology*, 14:78–106, 1982. [108](#)
- Esther Levin, Roberto Pieraccini, and Wieland Eckert. A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies. *IEEE Transactions on Speech and Audio Processing*, 8:11–23, 2000. [21](#), [83](#)
- Diane Litman, Michael Kearns, Satinder Singh, and Marilyn Walker. Automatic optimization of dialogue management. In *Proceedings of the 18th Conference on Computational Linguistics (COLING)*, Saarbruecken, Germany, 2000. [84](#)
- Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995. [47](#), [209](#)
- Xingkun Liu, Verena Rieser, and Oliver Lemon. A Wizard-of-Oz interface to study Information Presentation strategies for Spoken Dialogue Systems. In *Proceedings of the 1st International Workshop on Spoken Dialogue Systems (IWSDS)*, Irsee, Germany, 2009. [23](#)

REFERENCES

- François Mairesse, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Phrase-Based Statistical Language Generation Using Graphical Models and Active Learning. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics (ACL)*, Uppsala, Sweden, 2010. [31](#), [105](#)
- Bronislaw Malinowski. The Problem of Meaning in Primitive Languages. In Charles Kay Ogden and Ivor Armstrong Richards, editors, *The Meaning of Meaning*. London: Routledge and Keagan Paul, 1923. [1](#), [2](#)
- William Mann and Christian M I M Matthiessen. NIGEL: A systemic grammar for text generation. Technical report, ISI/RR-85-105, Information Sciences Institute, 1983. [13](#)
- William Mann and Sandra Thompson. Rhetorical Structure Theory: Towards a Functional Theory of Text Organisation. *TEXT*, 8(3):243–281, 1988. [15](#)
- Tomasz Marciniak and Michael Strube. Classification-based Generation Using TAG. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG)*, New Forest, England, UK, 2004. [4](#), [26](#)
- Tomasz Marciniak and Michael Strube. Beyond the Pipeline: Discrete Optimization in NLP. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL)*, Ann Arbor, 2005. [26](#), [34](#)
- Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. [116](#)

REFERENCES

- Colin Matheson, Massimo Poesio, and David Traum. Modelling Grounding and Discourse Obligations Using Update Rules. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Seattle, USA, 2000. [91](#)
- David Daniel McDonald. *Natural Language Production as a Process of Decision-Making Under Constraints*. PhD thesis, Massachusetts Institute of Technology (MIT), 1976. [15](#)
- Kathleen R McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cabridge University Press, Cambridge, 1985. [13](#)
- Igor Mel'cuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, New York, 1988. [13](#)
- Marie Meteer. Bridging the Generation Gap Between Text Planning and Linguistic Realization. *Computational Intelligence*, 7(4):296–304, 1991. [4, 25](#)
- Marie Meteer, David McDonald, Scott Anderson, David Forster, Linda Gay, Alison Huettner, and Penelope Sibun. MUMBLE-86: Design and Implementation. In *COINS*, University of Massachusetts, 1987. [15](#)
- Tom Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. [25](#)
- Sebastian Möller, Paula Smeele, Heleen Boland, and Jan Krebber. Evaluating Spoken Dialogue Systems according to Standards: A Case Study. *Computer, Speech and Language*, 21(1):26–53, 2007. [160](#)

REFERENCES

Johanna D. Moore and Cecile L. Paris. Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information. *Computational Linguistics*, 19:651–694, 1994. [15](#)

Reinhard Moratz, Kerstin Fischer, and Thora Tenbrink. Cognitive Modelling of Spatial Reference for Human-Robot Interaction. *International Journal on Artificial Intelligence Tools*, 10(4):589–611, 2001. [90](#)

Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD Thesis, UC Berkeley, Computer Science Division, 2002. [123](#)

Crystal Nakatsu and Michael White. Learning to Say It Well: Reranking Realizations by Predicted Synthesis Quality. In *In Proceedings of the Annual Meeting of the Association for Computational Linguistics (COLING-ACL) 2006*, pages 1113–1120, Sydney, Australia, 2006. [27](#)

Nicolas Nicolov, Chris Mellish, and Graeme Richie. Approximate Generation from Non-Hierarchical Representations. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux Castle UK, 1996. [13](#)

Jon Oberlander, Michael O’Donnell, Alistair Knott, and Chris Mellish. Conversation in the museum: experiments in dynamic hypermedia with the intelligent labeling explorer. *New Review of Hypermedia and Multimedia*, 4:11–32, 1998. [13](#)

Michael O’Donnell. Input Specification in the WAG Sentence Generation System. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux Castle UK, 1996. [13](#)

REFERENCES

- Alice Oh and Alexander Rudnicky. Stochastic Language Generation for Spoken Dialogue Systems. In *Proceedings of the ANLP/NAACL Workshop on Conversational Systems*, pages 27–32, Seattle, Washington, USA, 2000. [17](#)
- Nuria Olivier and Eric Horvitz. A Comparison of HMMs and Dynamic Bayesian Networks for Recognizing Office Activities. In *Proceedings of the 10th International Conference on User Modelling (UM'05)*, Edinburgh, Scotland, 2005. [123](#)
- Christos Papadimitriou and John Tsitsiklis. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3), 1987. [47](#), [209](#)
- Cécile Paris, Mingfang Wu, Keith Vander Linden, Matthew Post, and Shijian Lu. Myriad: An Architecture for Contextualized Information Retrieval and Delivery. In *IN AH2004: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH-2004)*, pages 205–214, 2004. [13](#)
- Cécile L. Paris. *User modelling in text generation*. Pinter Publishers, London, 1993. [13](#)
- Terry Patten. *Systemic Text Generation as Problem Solving*. Cambridge University Press, Cambridge, England, 1988. [15](#)
- Martin Pickering and Simon Garrod. Toward a mechanistic psychology of dialog. *Behavioral and Brain Sciences*, 27:169–226, 2004. [2](#), [106](#), [107](#)
- Martin Pickering and Simon Garrod. Alignment as the Basis for Successful Communication. *Research on Language and Computation*, 14(2):203–228, 2006. [xxiii](#), [108](#)

REFERENCES

- Martin J Pickering and Simon Garrod. Establishing and Using Routines During Dialogue: Implications for Psychology and Linguistics. In Ann Cutler, editor, *Twenty-First Century Psycholinguistics: Four Cornerstones*, pages 85–101. Erlbaum, Mahwah, NJ, 2005. [107](#)
- Olivier Pietquin and Dutoit. A Probabilistic Framework for Dialogue Simulation and Optimal Strategy Learning. *IEEE Transactions on Speech and Audio Processing*, 14(2):589–599, 2006. [21](#)
- Olivier Pietquin and Helen Hastie. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, Cambridge University Press, to appear. [135](#)
- Paul Piwek and Kees van Deemter. Generating under Global Constraints: The Case of Scripted Dialogue. *Research on Language and Computation*, 5(2):237–263, 2007. [15](#)
- Marcelo Quinderé, Luís Seabra Lopes, and António J. S. Texeira. An Information State Based Dialogue Manager for a Mobile Robot. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Antwerp, Belgium, 2007. [91](#)
- Lawrence Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 1989. [118, 123](#)
- Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1993. [118, 123](#)
- David Nicolás Racca, Luciana Benotti, and Pablo Duboue. The GIVE-2.5 C

REFERENCES

- generation system. In *Proceedings of the Generation Challenges Sessions at the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [165](#), [167](#)
- Adwait Ratnaparkhi. Trainable Methods for Surface Natural Language Generation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 194–201, 2000. [17](#)
- Ehud Reiter. Has a Consensus NL Generation Architecture Appeared and is it Psycholinguistically Plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, 1994. [3](#), [4](#), [28](#), [29](#)
- Ehud Reiter and Robert Dale. Building Applied Natural Language Generation Systems. *Natural Language Engineering*, 3(1):57–87, 1997. [xxi](#), [5](#)
- Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000. [xxi](#), [1](#), [3](#), [5](#)
- Ehud Reiter and Chris Mellish. Using Classification to Generate Text. In *Proceedings of the 9th International Conference on Computational Linguistics (COLING)*, Nantes, France, 1992. [13](#)
- Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 2006. [123](#)
- Verena Rieser and Oliver Lemon. Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz Data: Bootstrapping and Evaluation. In *Proceedings the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, Columbus, OH, USA, 2008. [24](#)

REFERENCES

- Verena Rieser and Oliver Lemon. Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. In *Proceedings the European Chapter of the Association for Computational Linguistics (EACL)*, Athens, Greece, 2009. [23](#)
- Verena Rieser and Oliver Lemon. *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Book Series: Theory and Applications of Natural Language Processing, Springer, 2011. [21](#)
- Verena Rieser, Oliver Lemon, and Xingkun Liu. Optimising Information Presentation for Spoken Dialogue Systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, 2010. [22](#), [23](#)
- Verena Rieser, Simon Keizer, Xingkun Liu, and Oliver Lemon. Adaptive Information Presentation for Spoken Dialogue Systems: Evaluation with human subjects. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [23](#)
- Robert Ross and John Bateman. Daisie: Information State Dialogues for Situated Systems. In *Proceedings of the 12th International Conference on Text, Speech and Dialogue (TSD)*, Pilsen, Czech Republic, 2009. [91](#)
- Michael Roth and Anette Frank. Computing EM-based Alignments of Routes and Route Directions as a Basis for Natural Language Generation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling)*, Beijing, China, 2010. [18](#)

REFERENCES

- Gavin A. Rummery and Mahesan Niranjan. *Online Q-Learning using connectionist systems*. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994. [46](#)
- Jost Schatzmann and Steve Young. The Hidden Agenda User Simulation Model. *IEEE Transactions on Speech and Audio Processing*, 17(4):733–747, 2009. [21](#)
- David Schlangen and Gabriel Skantze. A General, Abstract Model of Incremental Dialogue Processing. *Dialogue & Discourse*, 2(1):83–111, 2011. [192](#)
- Donia Scott and Johanna Moore. *An NLG evaluation competition? Eight Reasons to be Cautious*. Technical Report 2006/9, Department of Computing. The Open University, 2006. [83](#)
- Daniel Shapiro and Pat Langley. Separating skills from preference: Using learning to program by reward. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 570–577, 2002. [22](#)
- Stuart Shieber, Gertjan van Noord, Fernando Pereira, and Robert Moore. Semantic-head driven Generation. *Computational Linguistics*, 1(16):30–42, 1990. [13](#)
- Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, 16:105–133, 2002. [21, 84](#)
- Gabriel Skantze and Anna Hjalmarsson. Towards Incremental Speech Generation

REFERENCES

in Dialogue Systems. In *Proceedings of the 11th Annual Meeting on Discourse and Dialogue (SIGdial)*, Tokyo, Japan, 2010. [192](#)

Gabriel Skantze and David Schlangen. Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Athens, Greece, 2009. [192](#)

Burrhus Frederic Skinner. *The behavior of organisms*. New York: Appleton-Century-Crofts, 1938. [39](#)

Mark Steedman. *The Syntactic Process*. Cambridge, MA: MIT Press, 2000. [112](#)

Laura Stoia, Darla Shockley, Donna Byron, and Eric Fosler-Lussier. Noun phrase generation for situated dialogs. In *Proceedings of the 4th International Conference on Natural Language Generation (INLG)*, Sydney, Australia, 2006. [17](#), [82](#), [85](#)

Matthew Stone and Christine Doran. Paying Heed to Collocations. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux Castle UK, 1996. [13](#)

Matthew Stone and Bonnie Webber. Textual Economy through Close Coupling of Syntax and Semantics. In *Proceedings of the International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada, 1998. [25](#)

Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. Microplanning with Communicative Intentions: The SPUD System. *Computational Intelligence*, 19:311–381, 2001. [16](#)

REFERENCES

- Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. Textual Economy through Close Coupling of Syntax and Semantics. In *ACM Transactions on Graphics 23(3) (SIGGRAPH)*, pages 506–513, Los Angeles, CA, USA, 2004. [27](#)
- Kristina Striegnitz, Alexandre Denis, Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Mariet Theune. Report on the *Second* second challenge on generating instructions in virtual environments (GIVE-2.5). In *Proceedings of the Generation Challenges Sessions at the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [145](#), [162](#), [183](#)
- Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, pages 1038–1044. MIT Press, Cambridge, Massachusetts, 1996. [46](#)
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press, 1998. [xxi](#), [21](#), [41](#), [42](#), [44](#), [46](#), [74](#), [98](#)
- Csaba Szepesvari. *Algorithms for Reinforcement Learning*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, 2010. ISBN 9781608454921. [21](#)
- John Tsitsiklis. Asynchronous stochastic approximation and Q-Learning. *Machine Learning*, 16(3), 1994. [46](#)
- Tom Tullis and Bill Albert. *Measuring the User Experience: Collecting, Analyzing and Presenting Usability Metrics*. Morgan Kaufman, 2008. [69](#)

REFERENCES

- Kees van Deemter. What Game Theory Can Do for NLG: The Case of Vague Language. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG)*, Athens, Greece, 2009. [21](#)
- Menno van Zaanen. Bootstrapping Syntax and Recursion using Alignment-Based Learning. In *Proceedings of the 7th International Conference on Machine Learning (ICML)*, Stanford, California, USA, 2000. [113](#)
- Henriette Anna Elisabeth Viethen. *The Generation of Natural Descriptions: Corpus-Based Investigations of Referring Expressions in Visual Domains*. PhD Thesis, Macquarie University, Sydney, Australia, 2011. [83](#), [86](#)
- Jette Viethen, Robert Dale, and Markus Guhe. The Impact of Visual Context on the Content of Referring Expressions. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy, France, 2011. [83](#)
- Adam Vogel and Dan Jurafsky. Learning to Follow Navigational Directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, 2010. [19](#)
- Robert Wagner and Michael Fischer. The String-String Correction Problem. *Journal of the ACM*, 21(1):168–173, 1974. [113](#)
- Marilyn Walker. An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email. *Journal of Artificial Intelligence Research (JAIR)*, 12:387–416, 2000. [21](#)
- Marilyn Walker, Diane Litman, Candace Kamm, and Alicia Abella. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In *Proceedings of the*

REFERENCES

- 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Madrid, Spain, 1997. [68](#), [73](#), [141](#), [172](#)
- Marilyn Walker, Candace Kamm, and Diane Litman. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377, 2000. [xxix](#), [68](#), [70](#), [172](#), [193](#)
- Marilyn Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. Individual and Domain Adaptation in Sentence Planning for Dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 30:413–456, 2007. [4](#), [18](#), [29](#)
- Nigel Ward. *A Connectionist Language Generator*. Ablex, 1994. [13](#)
- Chris Watkins. *Learning from Delayed Rewards*. PhD Thesis, King's College, Cambridge, UK, 1989. [45](#), [46](#)
- Michael White, Robert Clark, and Johanna Moore. Generating Tailored, Comparative Descriptions with Contextually Appropriate Intonation. *Computational Linguistics*, 36(2):159–201, 2010. [13](#)
- Graham Wilcock and Yuji Matsumoto. Head-driven Generation with HPSG. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, 1998. [13](#)
- Deanna Wilkes-Gibbs and Herbert Clark. Coordinating beliefs in conversation. *Journal of Memory and Language*, 31:183–194, 1992. [107](#)
- Jason Williams. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *Proceedings of the Annual Conference of the International*

REFERENCES

Speech Communication Association (INTERSPEECH), Brisbane, Australia, 2008. [84](#)

Jason Williams and Steve Young. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(2):393–422, 2007. [21](#), [194](#)

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufman, 2005. [67](#), [86](#)

Maria Wolters, Kalliroi Georgila, Johanna Moore, Robert Logie, and Sarah MacPherson. Reducing Working Load Memory in Spoken Dialogue Systems. *Interacting with Computers*, 21(4):276–287, 2009. [160](#)

Sina Zarriess and Jonas Kuhn. Combining Referring Expression Generation and Surface Realization: A Corpus-Based Investigation of Architectures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria, 2013. [28](#)