Exploring Context-Sensitivity in Spatial Intention Recognition

Peter Kiefer and Christoph Schlieder

Laboratory for Semantic Information Technologies
Otto-Friedrich-University Bamberg
96045 Bamberg, Germany
{peter.kiefer,christoph.schlieder}@wiai.uni-bamberg.de

Abstract. In its most general form, the problem of inferring the intentions of a mobile user from his or her spatial behavior is equivalent to the plan recognition problem which is known to be intractable. Tractable special cases of the problem are therefore of great practical interest. Using formal grammars, intention recognition problems can be stated as parsing problems in a way that makes the connection between expressiveness and complexity explicit. We argue that context free grammars are not sufficiently expressive to handle important use cases. Furthermore, we identify three types of constraints on the grammar's productions that may arise in spatial intention recognition: rule-at-location constraints, rule-rule-constraints, and complex rule-location-constraints. Finally we show that Tree Adjoining Grammars can be used to handle rule-rule-constraints.

1 Introduction

Intention recognition is the problem of inferring a person's intentions to act from observations of that person's behavior. It can undoubtedly make an information system more intelligent. The system should automatically provide services that support the user's intentions, thus minimizing the necessary human computer interaction (information push). Especially applications used under conditions where the possibilities of human computer interaction are rather limited (such as biking or technical maintenance) should adapt to the limited cognitive resources of a user [1].

The intention recognition problem is also known in the literature as plan recognition problem [2]. Many current plan recognition approaches deal with the intractable form of the problem, and are thus rather suited for running on a server than on a stand-alone mobile client. We focus on the latter since we believe that there is a need for stand-alone mobile applications due to still expensive data tariffs and privacy issues.

Assigning intentions to a sequence of incoming behaviors can be interpreted as a pattern recognition problem. A formalism that has proven well for pattern recognition in areas like machine vision ([3, 4]) are Context Free Grammars (CFG) and their probabilistic counterparts. In addition to polynomial parsing

algorithms, such production systems have the advantage of being modular, generative, and intuitive.

Grammatical approaches are also used for intention recognition, but current research in this area agrees that the context-freeness of CFGs is not sufficiently expressive for most intention recognition domains. It seems at least a contradiction in terms to build a system that is context *aware* with a formalism that is context *free*. *Probabilistic State Dependent Grammars* (PSDG) ([5]), for instance, define the probability of a production rule as dependent on a general context variable (state) which is stored additional to the parse tree. This approach is very general and directed towards a large class of plan recognition problems which makes inference not efficient enough for our problem. *Spatially Grounded Intentional Systems* (SGIS) ([6]) include spatial context by making the applicability of a rule dependent on the current spatial region. This specific focus on spatial context makes this approach interesting for stand-alone mobile intention recognition of spatio-temporal behavior. A third line of argumentation targets at utilizing grammar formalisms with more context sensitivity from natural language processing (NLP) ([7]). *Mildly Context Sensitive Grammars* (MCSGs), a class of grammatical formalisms well-known in NLP since the 1980s, are proposed due to structural similarities to intention recognition problems. MCSGs are more expressive than CFGs while still being polynomially parsable which makes them especially attractive for algorithms running on mobile devices with low computational power.

This paper strengthens the argument made in [7] and tries to combine it with the spatial constraints formulated in SGIS [6]. We formulate three kinds of constraints on behaviors typical for intention recognition on spatio-temporal trajectories of a moving agent. This approach is new, in that it combines MCSGs, plan/intention recognition and spatial knowledge modeled by a domain expert. The goal consists in finding a formalism that has enough expressive power needed for the kind of context sensitivity imposed by our domain, while restricting the complexity of inference through spatial knowledge.

The outline of this paper is as follows. Section 2 introduces intention recognition with Context Free Grammars (CFG) and the problem of ambiguity. It is shown how spatial knowledge, like that encoded in SGIS, can help to resolve these ambiguities. Section 3 discusses the need for more context sensitive representations with special consideration of spatial knowledge. An example drawn from a location-based game is formalized using Tree Adjoining Grammars (TAG) which fall in the class of MCSGs. Section 4 sketches connections to related work in intention recognition and motion pattern analysis not mentioned in sections before. This paper describes work in progress, so we finally outline issues for our future work in Section 5.

Production Rules P			
$Trip$	$\rightarrow$	$Pick$ driving $Drop$	(1)
$Pick$	$\rightarrow$	parking	(2)
	|	parking driving parking	(3)
$Drop$	$\rightarrow$	parking	(4)
	|	parking driving parking	(5)

Start	**Destination**	**Observed Behavior Sequence**
$P_S \neq P_1$	$P_D \neq P_2$	parking driving parking driving parking driving parking
$P_S = P_1$	$P_D \neq P_2$	parking driving parking driving parking
$P_S \neq P_1$	$P_D = P_2$	parking driving parking driving parking
$P_S = P_1$	$P_D = P_2$	parking driving parking

Fig. 1. A very simple context free production system for intention recognition (top) and the language it defines (bottom).

2 Reducing Ambiguity by Adding Spatial Knowledge

2.1 Ambiguity: A Very Simple Example

To illustrate the basic ideas of this section we refer to the following simple example: suppose we observe an agent on a car trip from P_S to P_D. We know that she is going to pick up a friend at some point P_1 on that trip, and that she is going to drop her passenger at some other point P_2. The observations we get from our sensors are the behaviors $B = \{parking, driving\}$ while the concrete events of getting on or off the car remain hidden. The behaviors B are used as terminals in our context free production system. We want to decide for any element in the behavior sequence whether the driver is still on the way to the pick up point P_1 (having the intention $Pick$), or already accompanied by her friend (having the intention $Drop$). A context free production system sufficiently expressive for this simple use case is listed in Figure 1 (top). A top level intention $Trip$ is introduced as starting symbol, which yields in a set of intentions (non-terminals) $I = \{Trip, Pick, Drop\}$. Note that the two agents might have the same starting point ($P_S = P_1$), or the same destination ($P_D = P_2$). In Figure 1 (top), this is expressed by rules (2) and (4). Rule (2), for instance, states that the start of the journey and the pick up may coincide. Figure 1 (bottom) presents the language defined by our context free production system, i.e. all possible behavior sequences that can be created with rules (1)–(5). From an intention recognition perspective, these are all behavior sequences that can be recognized by the simple grammar. The first two columns indicate which of the four possible cases is covered by the accordant line, with respect to coinciding start/pick-up and destination/drop points.

Formally, we now have defined all necessary components of a Context Free Grammar $CFG_{simple} = (B, I, P, Trip)$, which we may also call an *intentional system* [6]. Recognizing the agent's intentions in this formalism consists in determining the correct parse tree for a sequence of behaviors. Algorithms for

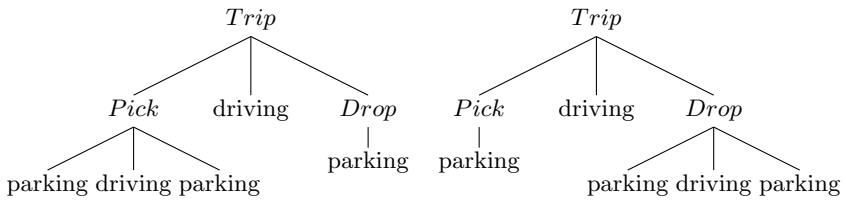

Fig. 2. Ambiguity: two possible parse trees for the same sequence of behaviors.

parsing CFGs, most based on chart-based parsers like the Earley Algorithm [8], are well-known and run in polynomial time. For each behavior b of an observation sequence, we can define the (direct) parent intention in the parse tree as the currently active intention. When building a context-aware service it is those active intentions that must be mapped to an appropriate information service. For instance, the third behavior in the parse trees in Figure 2 (parking) has an active intention *Pick* in the left tree, and *Drop* in the right tree.

The parse trees from Figure 2 visualize the ambiguity that may occur in our example. Without further knowledge we cannot decide which of the two trees is more plausible for the given behavior sequence. The problem of ambiguity is well-known in NLP and typically solved by assigning probabilities to the production rules, yielding in a Probabilistic Context Free Grammar (PCFG) (see [9] for an overview and examples). Rule probabilities for a PCFG in NLP are typically learned from large annotated text corpora. Assigning rule probabilities for an intention recognizing PCFG is much more complicated, especially because behavioral corpora do not exist for most domains. Instead of choosing a probabilistic approach, we rather try to discriminate between different behavior interpretations by using world knowledge, especially knowledge about the spatial structure of the environment, to assist the parsing process.

2.2 Disambiguation with Spatial Constraints

We have already introduced *intentional systems* which are CFGs with behaviors as terminals, and intentions as non-terminals. As a next step, we add spatial knowledge to the production rules and obtain *spatially grounded intentional systems* (SGIS), also defined in [6], as follows:

Definition 1. *Let R denote a set of spatial regions, B a set of behaviors, and I a set of intentions, all three sets being finite. A spatially grounded intentional system $A = (B, I, P, S, G)$ is an intentional system (B, I, P, S) together with a relation $G \subseteq P \times R$ describing the regions in which a production rule is applicable.*

SGISs exploit the fact that possible interpretations for an agent's behavior depend on the space he is currently located or moving in. This basic intuition of connection between intention and place has a long tradition and already appears

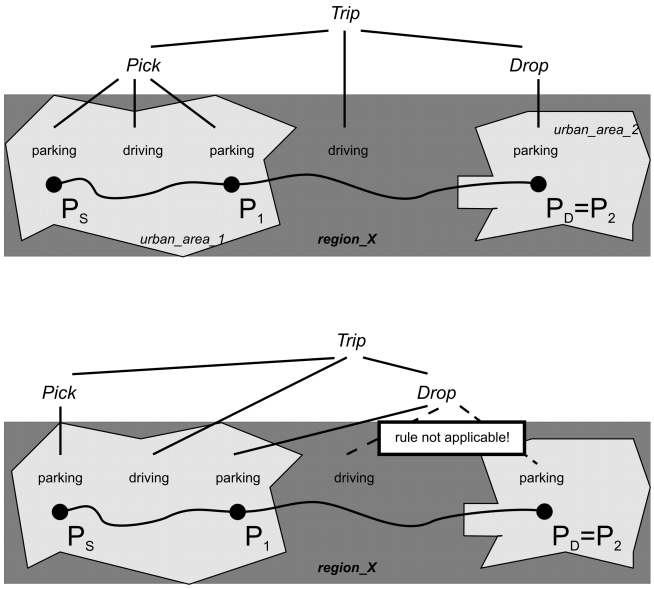

Fig. 3. Spatial disambiguation.

in the first article that has formulated the plan recognition problem [2] where the concept of $LOCATION$ was integrated as basic concept in the $World\ Domain$. For instance, a certain behavior in [2] can only be interpreted as plan $TAKE$, if both, the actor and the object of that possible $TAKE$–plan, are at the same location.

In our simple example, we could have some additional knowledge about the structure of space in which the two friends are traveling. For instance, they might be driving from $urban\_area\_1$ to $urban\_area\_2$ which are both part of $region_X$ (refer to Figure 3). We choose the spatial grounding of the production rules introduced in Figure 1 (top) as follows: rules (2) and (3) are grounded in region $urban\_area\_1$, rules (4) and (5) are grounded in region $urban\_area\_2$. Grounded means that the rule may only be applied if all of its basic behaviors (leaf nodes) take place in one of the selected areas. Thus, the production (1) must be grounded in all regions because the trip spans over all spatial regions from R.

Figure 3 shows how the spatial grounding in our simple example helps to disambiguate during parsing. For this given connection between behavior and space, the right parse tree of Figure 2 is impossible because rule (5) cannot be applied for regions outside of $urban\_area\_2$. Summarizing this section we can say that SGIS reduce the ambiguities that may occur when parsing a behavior sequence by modeling additional knowledge about the space where the behavior occurs.

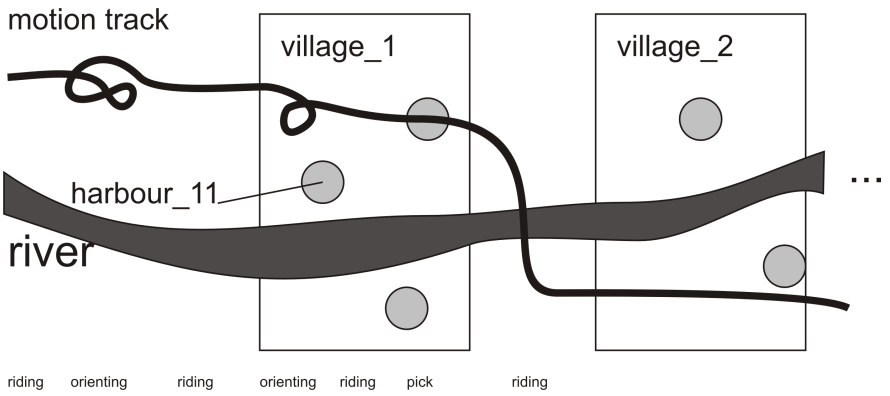

Fig. 4. Spatial behavior in the FluPa-Game.

3 Towards More Context Sensitivity

3.1 A Real World Example: The FluPa-Game

As in [6], we take a location-based game as use case for intention recognition. We may consider a location-based game "a game which is supported by localization technology and integrates the position of (one or several) players as main game element into its rules" [10]. In other words, players moving in a geographic gaming area are observed by localization technology, like GPS, which yields in a motion track as an input for intention analysis. The real advantage of choosing a game as use case are the limited possibilities of interaction. Especially when a game is played by bike, the user needs both hands for navigation. This is an ideal use case for implementing an information push mechanism.

A special case of location-based games are linear location-based games [11]. These are games where players are not free to move in the gaming area in an arbitrary manner, but follow a linear geographic feature, for instance a river or a railway line. This use case arises whenever the player's main intent consists in reaching a certain destination, like on a one-day bicycle tour from one city to another. The logics of such a game must ensure that players are not required to visit a location twice, for that would mean covering an extra distance. The first implemented, at least to our knowledge, bicycle game for linear trips was developed during the FluPa project at the Laboratory for Semantic Information Technologies, Otto-Friedrich-University Bamberg. The FluPa-Game has been implemented for Personal Digital Assistants (PDA). The GPS readings are taken from an external GPS receiver which transmits its data via Bluetooth. The implementation has just finished at the time of writing, so no user studies have been conducted yet.

We will use variations of the FluPa-Game in the rest of this section, so we need to sketch the main idea of the game concept: each of the two adversaries plays the part of a skipper on the river Regnitz (although, certainly, in reality

Production Rules for the FluPa-Game			
$PlayFluPa$	$\rightarrow$	$TreatVillage\ PlayFluPa$	(1)
	|	$SkipVillage\ PlayFluPa$	(2)
$SkipVillage$	$\rightarrow$	riding	(3)
$TreatVillage$	$\rightarrow$	riding $TreatVillage$	(4)
	|	riding $FindWay\ TreatVillage$	(5)
	|	$ChooseHarbour\ ReachHarbour\ ChangeGoods$	(6)
$FindWay$	$\rightarrow$	orienting	(7)
$ChooseHarbour$	$\rightarrow$	orienting	(8)
$ReachHarbour$	$\rightarrow$	riding	(9)
	|	riding $ReachHarbour$	(10)
	|	riding $FindWay\ ReachHarbour$	(11)
$ChangeGoods$	$\rightarrow$	picking	(12)
	|	dropping	(13)
	|	dropping picking	(14)

Fig. 5. Intentional system for the FluPa-Game.

they are moving by bike). It is basically a game around a transport optimization problem where you have to transport goods from one place to another to earn money. You have a certain capacity on your boat which you may not exceed with your freight. Before you may transport some goods, you must sign a delivery order of the form "4 sacks of grain from Eggolsheim to Forchheim for 16 gold pieces". The available delivery orders are optimized in a way that guarantees some competition between the players. Stations on the way are ordered linearly along the Regnitz, so you can only accept and execute delivery orders that lie ahead of you.

The types of spatial regions we can use for intention recognition are displayed in Figure 4: the regions of type *village* are ordered linearly along the river, and all part of the global region *game_area*. Each *village* contains at least two possible pick-up points, called *harbours*. All *harbours* inside one *village* are treated equal, in a sense that a player may reach any of them to trigger the same game action. An example intentional system for the intentions we want to support is given in Figure 5: a player decides before reaching a *village* if she wants to trigger some game actions there ($TreatVillage$) or just skip the village. Treating a village means: finding the village area, choosing one of the *harbours*, and going to the selected one (which again might include some wayfinding). Finally, the player must decide if to pick some goods, drop them, or do both actions. The motional behaviors are the result of a preprocessing of the GPS track which is segmented along the borders of the spatial regions (see [12] for details). The subscripts in Figure 4 give a hint of how a very simple preprocessing could look like. Picking and dropping events are taken directly from the user input (typing on the PDA).

Like in the simple example of section 2, we also have ambiguity for this grammar. For instance, the behavior sequence *riding, orienting, riding, orienting, riding, picking* is ambiguous, due to the two possible assignments of $FindWay$

Production Rules			
S	$\rightarrow$	$NP\ VP$	(1)
NP	$\rightarrow$	$Pronoun$	(2)
VP	$\rightarrow$	$Verb$	(3)
$Pronoun$	$\rightarrow$	I	(4)
	|	He	(5)
$Verb$	$\rightarrow$	live	(6)
	|	lives	(7)

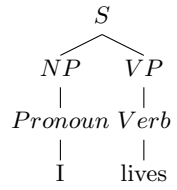

Fig. 6. A simple CFG from NLP (left) and one tree generated by this CFG (right).

(find way to the *village* or to the *harbour*). We will not present the corresponding parse trees here. It is also easy to find an appropriate spatial grounding for the rules that can help us in this example. Our point is that, through the rules of this specific game, we can formulate even more constraints on the applicability of certain production rules than the spatial grounding constraints.

3.2 A Parallel to NLP: Cross-Dependency Constraints

Most standard textbooks in NLP elaborate on syntax processing using CFGs like that shown in Figure 6 (left), which is certainly a very simplified one. Typical for NLP are the so-called preterminals which are nonterminals that can only have one terminal as child. These are typically used for word classes. For instance, $Pronoun$ and $Verb$ in Figure 6 are preterminals. Especially for preterminals, natural languages pose congruency constraints that are not covered by CFGs. The parse tree in Figure 6 (right), for instance, was deliberately chosen to generate a non-grammatical sentence. If the pronoun is chosen as "I", the verb must certainly be "live". This cannot be expressed in a CFG where the application of any production is by definition free of any other productions chosen for symbols in the tree that are not parent. In some natural languages, constraints like these can be even more complicated and show crossing dependencies such as in the pattern ABCABC.

In intention recognition, cross-serial dependencies like these can also occur. For intention recognition from spatio-temporal behaviors we can formulate three kinds of constraints, two of which may be cross-serial constraints:

1. **Rule-at-location constraints:** Certain productions may only be applied in certain regions.We described these constraints in section 2. These constraints are purely spatial constraints and covered by the SGIS formalism. They cannot be cross-serial because their influence is restricted to one subtree.
2. **Rule-rule constraints:** If a certain rule has been chosen, another rule must be chosen some time later in the parse tree. These constraints are equivalent to those in NLP and purely temporal constraints, because they do not pose any constraints on the regions where the rules are applied. These constraints can be cross-serial.

Production Rules			
$PlayFluPa$	$\rightarrow$	skip $PlayFluPa$	(1)
	|	pick $PlayFluPa$	(2)
	|	drop $PlayFluPa$	(3)

Fig. 7. Simplified intentional system for the FluPa-Game.

3. **Complex rule-location constraints**: If a certain rule has been applied in some region R, this restricts the freedom of applying another rule later in the parse tree. These constraints are spatial and temporal, and can also be cross-serial.

The examples we give for constraints 2 and 3 are drawn from the FluPa-Game. However, we do not need the full complexity of the rule set from Figure 5, because the most important aspect in this context is the choice between skipping a location, picking and dropping. Thus, for reasons of clarity, we use a simplified version of the FluPa grammar, as can be seen in Figure 7. Note that cross-dependencies appear just the same in the original full grammar from Figure 5 which we introduced for illustrative purposes.

A **rule-rule constraint** in the FluPa-Game is imposed by the fact that you cannot drop something before you have picked it somewhere else. In grammatical terms, that means that for any position in the input string, the number of 'drop' up to that point must not be greater than the number of 'pick'. It is obvious that this constraint can be handled with a non-deterministic pushdown automaton and thus be handled by a CFG. Thus, with a CFG we can avoid the application of rule (3) from Figure 7 when our stack (= the virtual boat in the game) is empty. However, intention recognition is not only about deciding if a certain input string belongs to our language, but rather about parsing the structure imposed by the rules. In other words, it does not suffice to decide that "pick pick drop pick drop drop" belongs to our language, but it will in many use cases also be important to associate the pairs of "pick" and "drop" that belong together. For instance, imagine "pick" and "drop" were only preterminals which in a second step had to be mapped to concrete goods, like "pick-grain" or "drop-fish". This long-ranging association between pairs cannot be expressed by a parse tree from a CFG.

A **complex rule-location constraint** in the FluPa-Game is given by the transport orders. For instance, if the player has picked the 4 sacks of grain in Eggolsheim (see the example transport order in section 3.1), the production that may be choosen in Forchheim is constrained to a "drop". One could also image typed constraints of the form: if you pick a sack of grain at a location of type *grain-pick-up-point*, you must choose the drop production at a location of type *grain-drop-point*.

3.3 Mildly Context Sensitive Grammars

Climbing up the well-known Chomsky hierarchy we come from regular grammars (type 3), over CFGs (type 2) to context-sensitive grammars (CSG, type 1). NLP

has extended this hierarchy by adding new levels between CSGs and CFGs. The first formalism in between were *Indexed Grammars* developed by Aho [13]. In this formalism, a stack is attached to each node in the parse tree and handed on to all children when a production is executed, after a possible push or pop operation. A restricted version of Indexed Grammars are *Linear Indexed Grammars* (LIG). They were first used, but not yet designated as LIG, in [14]. An important restriction of LIGs is that the stack is handed on only to one child node. This allows efficient parsing in polynomial time.

Other grammatical formalisms that fall in between CSGs and CFGs are *Tree Adjoining Grammars* (TAG, [15]), Head Driven Phrase Structure Grammars (HDPSG, [16]) and Combinatory Categorial Grammars (CCG, [17]). Interestingly, these other formalisms have been shown to be weakly equivalent to LIGs, i.e. their expressiveness allows them to produce the same languages ([18]). Thus, LIG, TAG, HDPSG, and CCG can be subsumed under a common class of language formalisms, called *Mildly Context Sensitive Grammars* (MCSG)[1]. Joshi [19] was the first to call this class MCSG and defined common properties: limited cross-serial dependencies, constant growth, and polynomial parsing.

The latter, polynomial parsing, obviously is an important argument for using MCSGs in our use case. Cross-serial dependencies can be important in many use cases as explained above. The various MCSG formalisms vary in the kind of cross-serial dependencies they support, for instance regarding to the number of symbols in a string that may be part of the same depend-relation[2]. The constant growth property is explained in [7] as: "this means that if there is a plan of length n then there is another plan of length at most n+K where K is a constant for the specific domain". This restrains the growth of the parsing sequence and is one main reason why these formalisms stay polynomially parsable. As the authors in [7], we see no use case in intention recognition that speaks against this property. For instance, an exponential growth in the possible length of plans cannot be observed in any reasonable domain.

3.4 Example: Reformulating the FluPa-Game as a TAG

Tree-Adjoining-Grammars (also called Tree-Adjunct-Grammars) are a MCSG formalism which is - different than other grammars - rather a tree generative system than a string manipulating system. That means, the elementary units that are manupulated by operations are not strings of terminals and nonterminals, but trees. An introduction to TAG can be found in [15]. In comparison to other MCFGs, TAGs are a rather intuitive way of modeling grammatical knowledge. Additionally, a parser for TAGS is available under GNU General Public License (XTAG, [20]).

The trees in a TAG are divided into initial trees and auxiliary trees (see Figure 8). Initial trees are headed by the starting symbol and have exclusively

[1] Geib and Steedman [7] also call them *LIG-equivalents*.

[2] Although in this paper we only present examples of two dependent symbols, one can easily imagine cases where three or more symbols in an input string share one depend-relation.

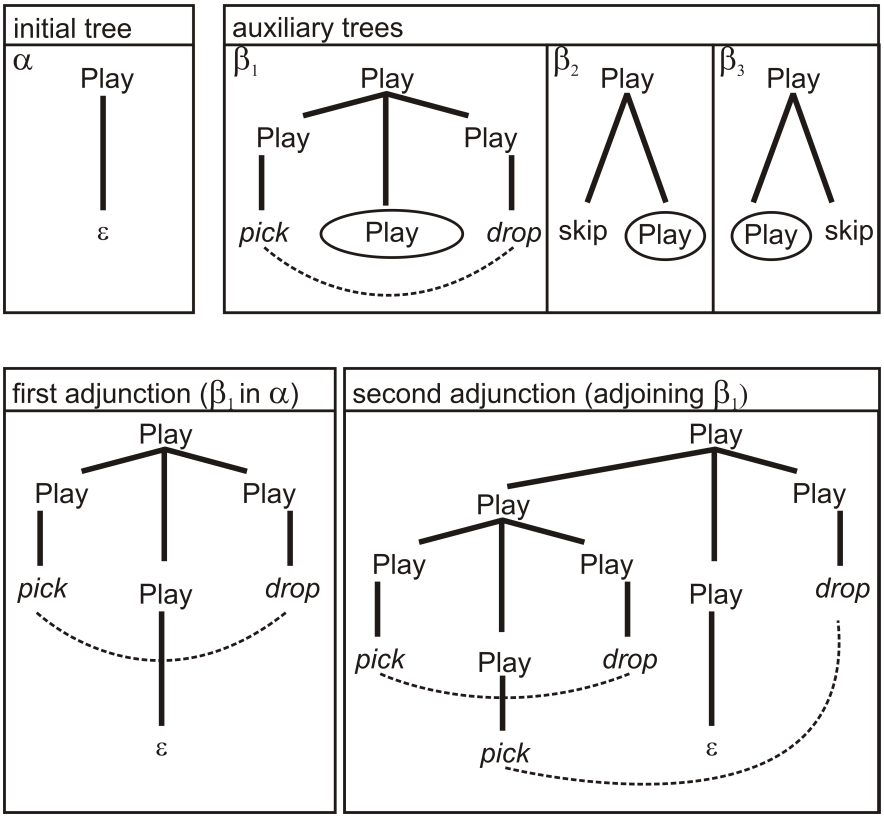

Fig. 8. A Tree-Adjoining-Grammar for the FluPa-Game (top) and two adjoining operations (bottom).

terminals in their leaf nodes. In other words: the initial tree itself already forms a correct word of the grammar through its terminals. Auxiliary trees have a head node, and a number of leaf nodes which are all but one terminal nodes. One of the leaf nodes is the so-called substitution node which has the same non-terminal as the head node (marked with a circle in Figure 8). On these trees, an operation called "adjunction" (or "adjoining") is defined. Through this operation, a non-terminal may be replaced by an auxiliary tree which has the same non-terminal as head and substitution node.

For instance, in the first step in Figure 8, the auxiliary tree β_1 is adjoined into the starting symbol *Play*. Thus, the original *Play* and ϵ from the initial tree are pushed down and appended to the substitution node. After the second adjunction, we get the final tree on the right bottom in Figure 8.

We have not mentioned the dotted lines yet: these are dependencies between nodes. TAGs allow these dependencies to be formulated on all initial and auxiliar trees. Through the adjoining operation, the dependency just gets stretched but

not destroyed. Thus, in our example TAG we can generate any cross-dependency of pick and drop just like needed in the FluPa-game use case. On these dependency links, different types of constraints can be formulated [21]. For instance, if pick and drop were pre-terminals, we could formulate a *selective adjoining constraint* stating that the drop pre-terminal must be terminalized to the corresponding symbol to the pick pre-terminal (e.g. pick-grain, drop-grain). This constraint is later used by the parsing algorithm.

4 Related Work

We have already cited [7] who recently made an argument for using MCSGs in plan recognition. As in our work, they argued with possible cross-dependencies in plan recognition. However, it was left open how exactly the correspondence between cross-dependencies in NLP and plan recognition can look like for specific domains. Adding spatial knowledge was also not addressed in that paper.

The plan recognition community has spent some interest in probabilistic network based approaches (Bayesian approaches, [22]). One recent work in this area chooses a Hierachical Markov Model and Particle Filtering to predict a user's changes in transportation mode, like getting on or off a bus [23]. They evaluate their methods using a system called Opportunity Knocks which aims at helping cognitively-impaired people to use public transportation safely. Different to our approach, their mobile client sends data to an inference engine on a server so that weaker demands on the computational complexity of the algorithm hold. Another probabilistic network based approach, the Abstract Hidden Markov Model, is chosen by [24]. Again, they evaluate their algorithms not on a mobile client, but in an intelligent office environment.

A probabilistic model is also used in [25], here with the Dempster-Shafer theory of evidential reasoning. This approach is interesting for it formalizes spatial knowledge as spatial conceptual maps. The system collects evidence for a set of candidate locations from which the most probable is chosen. The user's future trajectory is predicted, not his or her intention.

Inferring the goals of a moving human agent is also the aim of [26]. However, the focus here is on detecting "inexplicable behavior" in an observed environment (like a car park), not on explaining the high level intentions. Possible goals are restrained to "reaching a certain point P". Sufficient for these simple goals, a representational formalism with low expressiveness (a state-transition diagram) is chosen.

An ex-post analysis of spatio-temporal trajectories that have occurred in a RoboCup game is the use case of [27]. This paper is related to our approach in that the motion patterns are topologically contextualized by cutting them at the edges of spatial regions. However, not the changing intentions of an agent at a certain time are analyzed, but the overall characterization of a game, for instance "BalancedWingPlaying(TeamA) = true".

In the extended FluPa-Game example from Figure 5 we assumed a stream of incoming preprocessed behaviors, namely orienting and riding. These behav-

iors can be even much more complex, as in the museum example from [28] (e.g. crossing, visiting, ant-like-touring, grasshopper-like-touring). To obtain such behaviors, qualitative characteristics need to be extracted from an observed motion path (or motion segment). Methods for this task have been developed in the area of spatial cognition, e.g. [29].

The types of behaviors that are of interest for a specific domain can generally be obtained in two different ways: one way is to model them by hand by asking a domain expert. On the other hand, we could detect them from a set of observed trajectories, e.g. when a number of FluPa-Games has produced a collection of recorded tracks. The task of automatic motion pattern detection is one concern of the spatio-temporal data mining community, see for instance [30].

On the lowest preprocessing level, before we can even get any reliable motion paths, we must deal with sensor noise. Players in the FluPa-Game, for example, are positioned with imprecise GPS measurements. Although newest GPS chip sets deliver quite high precision and accuracy, the position might still be noisy in situations with few satellites in view, for instance in the forest. To solve this problem, Bayesian Filtering techniques may help us [31]. For our use case of processing everything on-device, the benefits of Bayesian Filtering and its costs need to be evaluated deliberately.

5 Discussion and Future Work

In this paper, we have argued that CFGs are not sufficiently expressive for many use cases of intention recognition of spatio-temporal behavior. We identified three kinds of constraints on productions rules that may occur. We have shown that with TAGs we can express rule-rule constraints, while with SGIS we can formalize rule-at-location constraints.

What still remains open is the representation of complex rule-location constraints. It is not yet clear if all complex constraints that can occur are expressible with the same formalism, or if this category needs a further refinement. Furthermore, other grammatical formalisms than TAG need to be considered. An interesting issue we have skipped in this paper are probabilistic grammars. Even with spatial grounding of production rules, ambiguity in an input sequence may occur. For these cases, adding probabilities to our grammar might help.

Finally, we plan to evaluate the use of MCFGs with spatial information in user studies with the FluPa-Game. Other use cases are also planned, for instance a mobile tourist guide.

Acknowledgements

We wish to thank Klaus Stein for the discussions on the problem of intention recognition. Further, we would like to thank the FluPa-team (http://www.kinf.wiai.uni-bamberg.de/flupa/) for their help with the game concept and implementation.

The FluPa-project was funded by the Wasserwirtschaftsamt Kronach (watershed management office Kronach) and the Umweltstation Lias-Grube Unterstürmig e.V. (office for environmental education in Unterstürmig).

References

1. Baus, J., Krueger, A., Wahlster, W.: A resource-adaptive mobile navigation system. In: Proc. 7th International Conference on Intelligent User Interfaces, San Francisco, USA, ACM Press (2002) 15–22
2. Schmidt, C., Sridharan, N., Goodson, J.: The plan recognition problem: An intersection of psychology and artificial intelligence. Artificial Intelligence **11**(1-2) (1978) 45–83
3. Chanda, G., Dellaert, F.: Grammatical methods in computer vision: An overview. Technical Report GIT-GVU-04-29, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA (2004) ftp://ftp.cc.gatech.edu/pub/gvu/tr/2004/04-29.pdf.
4. Ivanov, Y., Bobick, A.: Recognition of visual activities and interactions by stochastic parsing. IEEE transactions on pattern analysis and machine intelligence (PAMI) **22**(8) (2000) 852–872
5. Pynadath, D.V., Wellman, M.P.: Probabilistic state-dependent grammars for plan recognition. In: Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence. (2000) 507–514
6. Schlieder, C.: Representing the meaning of spatial behavior by spatially grounded intentional systems. In: GeoSpatial Semantics, First International Conference. Volume 3799 of Lecture Notes in Computer Science., Springer (2005) 30–44
7. Geib, C.W., Steedman, M.: On natural language processing and plan recognition. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI). (2007) 1612–1617
8. Earley, J.: An efficient context-free parsing algorithm. Communications of the ACM **13**(2) (1970) 94–102
9. Jurafsky, D., Martin, J.H., eds.: Lexicalized and Probabilistic Parsing. In: Speech and Language Processing. Prentice Hall, Upper saddle River, New Jersey, USA (2000) 447–476
10. Kiefer, P., Matyas, S., Schlieder, C.: Geogames - integrating edutainment content in location-based games. In Magerkurth, C., Roecker, C., eds.: Pervasive Games - Applications. (2007) to appear.
11. Kiefer, P., Matyas, S., Schlieder, C.: Playing on a line: Location-based games for linear trips. In: International Conference on Advances in Computer Entertainment Tecnology (ACE 2007). (2007) to appear.
12. Stein, K., Schlieder, C.: Recognition of intentional behavior in spatial partonomies. In: ECAI 2004 Worskhop 15: Spatial and Temporal Reasoning (16th European Conference on Artificial Intelligence). (2005)
13. Aho, A.V.: Indexed grammars - an extension of context-free grammars. Journal of the ACM **15**(4) (1968) 647–671
14. Gazdar, G.: Applicability of indexed grammars to natural languages. In: Natural Language Parsing and Linguistic Theories. D. Reidel, Dordrecht (1988)
15. Joshi, A.K., Schabes, Y.: Tree-adjoining grammars. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Volume 3. Springer, Berlin, New York (1997) 69–124

16. Sag, I.A., Pollard, C.: Head-driven phrase structure grammar: An informal synopsis. CSLI Report 87-79, Stanford University, Stanford University (1987)
17. Steedman, M.: Dependency and coordination in the grammar of dutch and english. Language **61**(3) (1985) 523–568
18. Vijay-Shanker, K., Weir, D.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory **27**(6) (1994) 511–546
19. Joshi, A.K.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In Dowty, D.R., Karttunen, L., Zwicky, A.M., eds.: Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives. Cambridge University Press, Cambridge (1985) 206–250
20. XTAG Research Group: A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania (2001)
21. Joshi, A.K.: Introduction to tree adjoining grammars. In Manaster-Ramer, A., ed.: Mathematics of Language, Amsterdam/Philadelphia, John Benjamins (1987) 87–114
22. Charniak, E., Goldman, R.P.: A bayesian model of plan recognition. Artificial Intelligence **64**(1) (1993) 53–79
23. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Learning and inferring transportation routines. Artificial Intelligence **171**(5-6) (2007) 311–331
24. Bui, H.H.: A general model for online probabilistic plan recognition. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). (2003)
25. Samaan, N., Karmouch, A.: A mobility prediction architecture based on contextual knowledge and spatial conceptual maps. IEEE Transactions on Mobile Computing **4**(6) (2005) 537–551
26. Dee, H., Hogg, D.: Detecting inexplicable behaviour. In: Proceedings of the British Machine Vision Conference, The British Machine Vision Association (2004) 477–486
27. Gottfried, B., Witte, J.: Representing spatial activities by spatially contextualised motion patterns. In Lakemeyer, G. et al., ed.: RoboCup 2007, International Symposium,, Springer (2007) 329–336
28. Schlieder, C., Werner, A.: Interpretation of intentional behavior in spatial partonomies. In: Spatial Cognition III, Routes and Navigation, Human Memory and Learning, Spatial Representation and Spatial Learning. Volume 2685 of Lecture Notes in Computer Science., Springer (2003) 401–414
29. Musto, A., Stein, K., Eisenkolb, A., Röfer, T., Brauer, W., Schill, K.: From motion observation to qualitative motion representation. In: Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications, LNCS 1849, London, UK, Springer (2000) 115–126
30. Laube, P., van Krefeld, M., Imfeld, S.: Finding remo - detecting relative motion patterns in geospatial lifelines. In: Developments in Spatial Data Handling, Proceedings of the 11th International Symposium on Spatial Data Handling. (2004) 201–215
31. Fox, D., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian filtering for location estimation. IEEE Pervasive Computing **2**(3) (2003) 24–33