

Spatially Constrained Grammars for Mobile Intention Recognition

Peter Kiefer

Laboratory for Semantic Information Technologies
Otto-Friedrich-University Bamberg
96045 Bamberg, Germany
`peter.kiefer@uni-bamberg.de`

Abstract. Mobile intention recognition is the problem of inferring a mobile agent's intentions from her spatio-temporal behavior. The intentions an agent can have in a specific situation depend on the spatial context, and on the spatially contextualized behavior history. We introduce two spatially constrained grammars that allow for modeling of complex constraints between space and intentions, one based on Context-Free, one based on Tree-Adjoining Grammars. We show which of these formalisms is suited best for frequently occurring intentional patterns. We argue that our grammars are cognitively comprehensible, while at the same time helping to prune the search space for intention recognition.

Keywords: Intention recognition, Mobile assistance systems.

1 Introduction

The problem of inferring an agent's intentions from her behavior is called *intention recognition problem*. The closely related problem of plan recognition has been discussed in AI literature since many years [1]. Approaches for plan recognition differ in the way the domain and possible plans are represented. While early work tended to be quite general, like Kautz's event hierarchies [2], current research is typically concerned with specialized use cases (e.g. [3]), and efficient inference (e.g. [4]).

A class of intention recognition problems with specific need for efficient inference is *mobile intention recognition*. We observe a mobile user's trajectory and try to 'guess' what intentions she has in mind. These mobile problems are different, not only because of the restricted computational and cognitive resources [5]. Mobile intention recognition problems also differ to 'traditional' use cases because mobile behavior happens in space. This has a number of implications. One is that we have knowledge about the spatial context, about spatial objects, their relations, and spatial constraints. A glance at current research on the inverse problem, spatio-temporal planning, gives us an idea how these constraints can look like: Seifert et al. discuss an interactive assistance system that supports in spatio-temporal planning tasks [6]. In their example they describe the constraints that need to be considered when planning a trip: the temporal order of activities,

the time needed for traveling from A to B, and spatial constraints about what actions can be performed at which location. Important about Seifert's approach is that the chosen hierarchical spatial structure offers a cognitively appealing way of interaction between user and planning system, while at the same time helping to prune the search space.

In this paper, we will see that complex constraints between intentions and space not only give us a rich toolbox to formalize typical behavioral patterns in mobile intention recognition, but can also speed up inference. We choose formal grammars to represent intentions so that the intention recognition problem becomes a parsing problem. Grammars are, in general, cognitively easy to understand and make the connection between expressiveness and complexity explicit. The main contribution of this paper is the combination of spatial constraints with *Tree Adjoining Grammars* (TAG), a formalism from natural language processing (NLP) that falls in complexity between context-free and context-sensitive grammars (CFG, CSG). The idea to apply grammar formalisms from NLP to plan/intention recognition is also followed by Geib and Steedman [7], and in our own previous work [8]. In difference to these approaches, our spatially constrained grammars allow the formalization of complex, non-local constraints between intentions and space (and not only between intentions).

The rest of this paper is structured as follows: in section 2 we explain which steps are necessary to state a mobile intention recognition problem as a parsing problem. In this context we review *Spatially Grounded Intentional Systems* (SGIS) [9]. In section 3, we explain which important use cases cannot be handled with SGIS, and proceed over *Spatially Constrained Context-Free Grammars* (SCCFG) to *Spatially Constrained Tree-Adjoining Grammars* (SCTAG). Using real motion track data from the location-based game CityPoker we discuss which general spatio-temporal behavior patterns are handled best with which formalism. The paper closes with a discussion of related work (section 4) and an outlook on questions that remain open (section 5).

2 From Spatio-temporal Behavior to Intentions

2.1 Mobile Intention Recognition

The fact that mobile behavior happens in space and time has mainly two implications: one is that we can take use of spatial information. We do not only know the absolute coordinate of a user's behavior, but also the spatial context. With an according spatial model we can say that the behavior happened, for instance, in a specific region, on a road, or close to a point of interest. We also have information about the spatial relations between these objects [10], like *intersect*, *overlap*, or *north_of*. Depending on the specific use case, these spatial objects also bear a certain semantics: '*a restaurant is a place where I can have the intention to eat something*'. This is very similar to the basic intuition behind activity-based spatial ontologies [11]. However, inferring the agent's intention directly from her position is too simple in many situations: a mobile user passing



Fig. 1. Segmented motion track with classified behavior sequence from a CityPoker game. (The player enters from the right.)

by a restaurant does not necessarily have the intention to eat there. Schlieder calls this spatio-temporal design problem *room crossing problem* [9].

This leads us to the second implication of spatio-temporality: the gap between sensor input (e.g. position data from a GPS device) and high-level intentions (e.g. ‘find a restaurant’) is extremely large. It is not possible to design an intelligent intention recognition algorithm that works *directly* on pairs of (latitude/longitude). To bridge this gap, we use a multi-level architecture with the level of *behaviors* as intermediate level between *position* and *intention*. We process a stream of (lat/lon)-pairs as follows:

1. *Preprocessing.* The quality of the raw GPS data is improved. This includes removing points with zero satellites, and those with an impossible speed.
2. *Segmentation.* The motion track is segmented at the border of regions, and when the spatio-temporal properties (e.g. speed, direction) of the last n points have changed significantly [12].
3. *Feature Extraction.* Each segment is analyzed and annotated with certain features, like speed and curvature [13].
4. *Classification.* Using these features, each motion segment is classified to one *behavior*. We can use any mapping function from feature vector to behaviors, for instance realized as a decision tree.

As output we get a stream of behaviors. In the example from Fig. 1 we distinguish the following behaviors: riding (b_r), standing (b_0), sauntering (b_s), curving (b_c), and slow-curving (b_{cs}). This track was recorded in the location-based game CityPoker. In this game, two players are trying to find (physical) playing cards

which are hidden in a city. The gaming area is structured by five rectangular cache regions. In each cache region there are three potential cache coordinates (one is drawn as a circle in Fig. 1). Cards are only hidden in one of the three potential caches. Players can find out about the correct cache by answering a multiple choice question. Once they have arrived at the cache, they perform a detail search in the environment, under bushes, trees, or benches, until they finally find the cards. They may then trade one card against one from their hand, and continue in the game. For a complete description of the game, refer to [9].

The reason why this game is especially suited as exemplary use case is that CityPoker is played by bike at high speed. The user's cognitive resources are bound by the traffic, and she does not have the possibility to interact with the device (a J2ME enabled smartphone, localized by GPS) in a proper way. Similar situations occur in other use cases, like car navigation or maintenance work. Depending on the intention recognized we want to select an appropriate information service automatically. For instance, if we recognize the intention *Find_Way* we will probably select a map service. It is up to the application designer to decide whether to present the service with information push, or just to ease the access to this service ('hotbutton'). We will not discuss the step of mapping intentions to information services any further in this paper.

2.2 Parsing Behavior Sequences

The stream of behaviors described above serves as input to a parsing algorithm. Using behaviors as terminals and intentions as non-terminals, we can write rules of a formal grammar that describe the intentions of an agent in our domain. Most plan recognition approaches have followed a hierarchical structure of plans/intentions (e.g. [14,15]). We should say something about the difference between *plans* and *intentions* although an elaborate discussion of this issue is beyond the scope of this paper. In line with common BDI agent literature, we see intentions as 'states of mind' which are directed 'towards some future state of affairs' ([16, p.23]). We see 'plans as *recipes* for achieving intentions.' [16, p.28]. We can say that a rule in our grammar describes a plan, while each non-terminal stands for one intention. Thus, the aim of intention recognition is to find out (at least) the current intention.

In CityPoker, for instance, a player will certainly have the intention to *Play*. At the beginning of each game, the members of a team discuss their strategy. Playing in CityPoker means exchanging cards in several cache regions, so we model a sequence of intentions as follows: *GotoRegion HandleRegion, GotoRegion HandleRegion*, and so on. In the cache region players find themselves a comfortable place to stand, answer a multiple-choice question, and select one out of three caches, depending on their answer. In the cache, they search a playing card which is hidden in the environment (see the behavior sequence in Fig. 1).

A context-free production system for CityPoker is listed in Fig. 2¹. Grammar rules like these are modular and intuitively understandable, also for non-computer scientists. Formal properties of grammars are well-known, and parsing

¹ Rules with a right-hand side of the form $(symbol_1|...|symbol_n)^+$ are a simplified notation for 'an arbitrary sequence of $symbol_1, \dots, symbol_n$, but at least one of them'.

algorithms exist. The choice of the formalism depends on the requirements of the use case. We briefly recall that with a CFG, we can express patterns of the form $a^n b^n$. As argued in [9], most intention recognition use cases need at least this expressiveness. A typical example is leaving the same number of regions as entered before ($enter^m leave^n$). Note that parsing a stream of behaviors must be done incrementally, i.e. with an incomplete behavior sequence. We can find the currently active intention in the parse tree by choosing the non-terminal which is direct parent of the current behavior.

2.3 Reducing Parsing Ambiguities by Adding Spatial Knowledge

When parsing formal grammars we easily find ourselves in a situation where the same input sequence may have two or more possible parse trees, i.e. more than one possible interpretation. This is especially true when parsing an incomplete behavior sequence incrementally. One way to deal with ambiguity are probabilistic grammars [17] where we have to determine a probability for each rule in the grammar. A spatial way of ambiguity reduction is proposed by Schlieder in [9]: SGIS are context-free production systems, like that in Fig. 2, with the extension that each rule is annotated with a number of regions in which it is applicable. We call this the *spatial grounding* of rules. For instance, a *HandleCache* intention is grounded in all regions of type *cache*. We modify all rules accordingly. An SGIS rule for the original rule (12) would look like follows:

$$\begin{aligned} \textit{HandleCache} &\rightarrow \textit{SearchCards} \textit{DiscussStrategy} \\ &[\textit{grounding} : \textit{cache}_{1,1}, \dots, \textit{cache}_{5,3}] \end{aligned}$$

This reduces the number of possible rules applicable at each position in the behavior sequence, thus avoiding many ambiguities. Figure 3 shows two possible interpretations for the behavior sequence from Fig. 1: without spatial knowledge we could not decide which of the two interpretations is correct. For parsing in SGIS we replace the pure behavior stream $(beh_1, beh_2, beh_3, \dots)$ by a stream of behavior/region pairs: $((beh_1, reg_1), (beh_2, reg_2), (beh_3, reg_3), \dots)$. Each behavior is annotated with the region in which it occurs. Also the non-terminals in the parse tree are annotated with a region (*Intention, region*), with the meaning that all child-intentions or child-behaviors of this intention must occur in that region. SGIS are a short form of writing rules of the following form (where *Symbol* can be an intention or a behavior):

$$(\textit{Intention}, reg_x) \rightarrow (\textit{Symbol}_1, reg_x) \dots (\textit{Symbol}_n, reg_x)$$

That means, we cannot write rules for arbitrary combinations of regions. In addition, we require that another rule can only be inserted at an intention \textit{Symbol}_i if the region of the other rule is (transitive) child in the partonomy, i.e. in the above rule we can only insert productions with a region $reg_y \textit{part_of} reg_x$ (which includes the same region: $reg_y.\textit{equals}(reg_x)$). SGIS have been designed for partonomially structured space. The nesting of rules follows closely the nesting of regions

Production Rules for CityPoker		
<i>Play</i>	\rightarrow	<i>DiscussStrategy Continue</i> (1)
<i>DiscussStrategy</i>	\rightarrow	b_0 (2)
<i>Continue</i>	\rightarrow	$\varepsilon \mid \textit{GotoRegion HandleRegion Continue}$ (3)
<i>GotoRegion</i>	\rightarrow	$(b_r \mid b_0 \mid b_c)^+$ (4)
<i>HandleRegion</i>	\rightarrow	<i>SelectCache GotoCache HandleCache</i> (5)
<i>SelectCache</i>	\rightarrow	<i>FindParkingPos AnswerQuiz</i> (6)
<i>FindParkingPos</i>	\rightarrow	$(b_r \mid b_c \mid b_{cs})^+$ (7)
<i>AnswerQuiz</i>	\rightarrow	b_0 (8)
<i>GotoCache</i>	\rightarrow	$(\textit{SearchWayToC} \mid \textit{NavigateTowardsC})^+$ (9)
<i>SearchWayToC</i>	\rightarrow	$(b_0 \mid b_{cs} \mid b_s)^+$ (10)
<i>NavigateTowardsC</i>	\rightarrow	$(b_r \mid b_c)^+$ (11)
<i>HandleCache</i>	\rightarrow	<i>SearchCards DiscussStrategy</i> (12)
<i>SearchCards</i>	\rightarrow	$(\textit{CrossCache} \mid \textit{DetailSearch})^+$ (13)
<i>CrossCache</i>	\rightarrow	$(b_r)^+$ (14)
<i>DetailSearch</i>	\rightarrow	$(b_0 \mid b_{cs} \mid b_s \mid b_c)^+$ (15)

Fig. 2. Context-free production rules for intention recognition in CityPoker

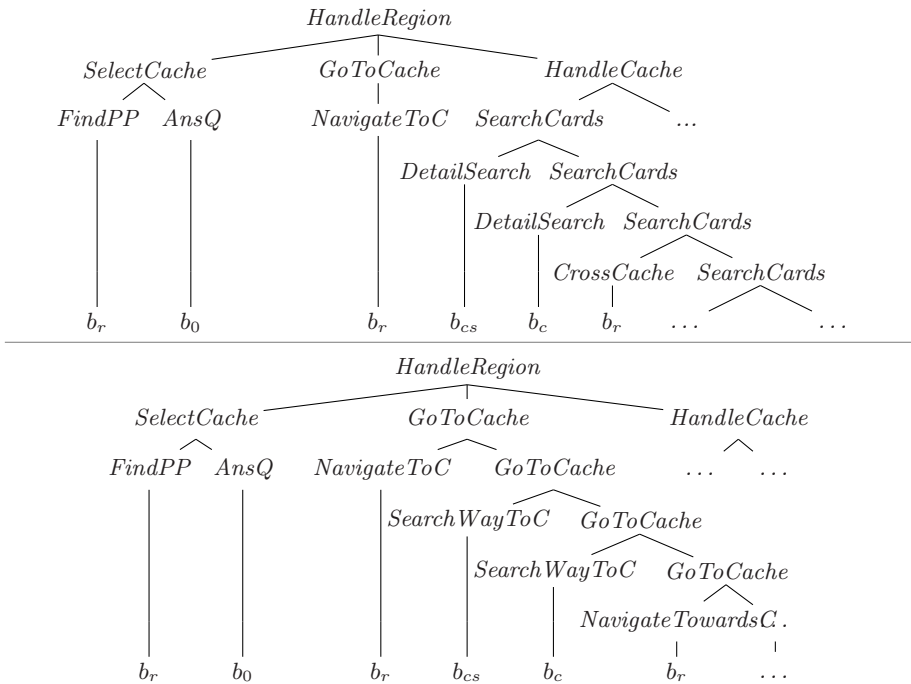


Fig. 3. Parsing ambiguity if we had no spatial knowledge (see track from Fig. 1). Through spatial disambiguation in SGIS we can decide that the bottom parse tree is correct.

and sub-regions in the spatial model. The CityPoker partonomy is structured as follows: the `game_area` contains five rectangular `cache_regions`, each of which in turn contains three caches.

SGIS deliberately restrict us in what we can express: we cannot write rules for arbitrary pairs of behavior and region. This makes sense from a spatial point of view (the agent cannot ‘beam’ herself), as well as from a cognitive point of view: as in Seifert et al. [6], the knowledge engineer is working with a representational formalism that resembles a structure of space preferred by many individuals: a hierarchical one [18].

3 Spatially Constrained Grammars

3.1 Spatially Constrained Context-Free Grammars

SGIS is a formalism with which we can model a variety of spatio-temporal intention recognition problems. With the spatial grounding of rules we can formalize spatial constraints of type *part_of*. Constraints about the temporal order of intentions are formalized implicitly through the order of right-hand symbols in the production rules.

However, the restrictions of SGIS hinder us from expressing frequently occurring use cases. Consider the motion track in Fig. 1: the agent enters the cache, shows some searching behavior, and then temporarily leaves the circular cache to the south. Knowing the whole motion track we can decide that this is better described as an accidental leaving, i.e. no intention change, than as a *ChangePlan* intention². For an incremental algorithm, it is not clear at the moment of leaving whether the agent will return. It is also not necessary that the intermediate behavior is located in the parent `cache_region` of the cache. Finally, entering just *any* cache is not sufficient for accidental leaving, but we require that cache to be the same as left before. We would need the following rule

$$\begin{aligned} (HandleCache, cache_{1,1}) \rightarrow & (SearchCards, cache_{1,1}), \\ & (accidental\ leaving\ behavior, [unconstrained]), \\ & (SearchCards, cache_{1,1}) \end{aligned}$$

We cannot formulate this in SGIS, but still it makes no sense to write rules for pairs of (intention, region). We have already argued against this maximum of complexity in section 2.3. At this point, we can add another argument: we would have to write a plethora of similar rules for each cache in our game. What we would need to formalize the *accidental leaving pattern* elegantly is the following:

$$HandleCache \rightarrow \overset{\textit{identical}}{\textit{SearchCards Confused SearchCards}}$$

² A player in CityPoker who has given a wrong answer to the quiz will be searching at the wrong cache and probably give up after some time. He will then head for one of the other caches. The *ChangePlan* intention was omitted in Fig. 2 for reasons of clarity.

We can easily find other examples of the pattern ‘a certain behavior/intention occurs in a region which has a spatial relation *r* to another region where the agent has done something else before’. For instance, we can find use cases where it makes sense to detect a *ReturnToX* intention if the agent has forgotten the way back to some place. We could define this as ‘the agent shows a searching behavior in a region which *touches* a region she has been to before’:

$$\text{ClothesShopping} \rightarrow \text{ExamineClothes} \overset{\textit{touches}}{\text{HaveABreak ReturnToShop}}$$

The definition of a new spatial context-free grammar that handles these examples is quite straightforward.

Definition 1. A *Spatially Constrained Context-Free Grammar* is defined as $\text{SCCFG} = (\text{CFG}, R, \text{SR}, \text{GC}, \text{NLC})$, where

- *CFG* is a context-free grammar (I, B, P, S) , defined over intentions *I*, and behaviors *B*, with production rules *P* and start symbol *S* (the top-level intention).
- *R* is a set of regions
- *SR* is a set of spatial relations, where each relation $r \subseteq R \times R$
- $\text{GC} \subseteq P \times R$ is a set of grounding constraints (as in *SGIS* [9])
- *NLC* is a set of spatial non-local constraints. Each constraint has a type from the spatial relations *SR* and is defined for two right-hand symbols of one production rule from *P*.

We introduce the grounding constraints to make *SCCFG* a real extension of *SGIS*. However, we will not always need them, as in the *CityPoker* example. The reason is that *CityPoker*-regions are typed according to their level in the partonomy (cache *part_of* cache_region *part_of* gameboard). With a *SCCFG* we can rewrite the rules from Fig. 2 without spatial grounding in a specific region, but with *part_of* and *identical* relations, for instance for rules (5) and (12):

$$\begin{aligned} \text{HandleRegion} &\rightarrow \text{SelectCache} \overset{\textit{identical}}{\text{GotoCache}} \overset{\textit{part_of}}{\text{HandleCache}} \\ \text{HandleCache} &\rightarrow \text{SearchCards} \overset{\textit{identical}}{\text{DiscussStrategy}} \end{aligned}$$

SCCFG obviously have a higher expressiveness than *SGIS*. We can express more spatial relations than *part_of*, and create a nesting of relations by applying the production rules. In difference to *SGIS*, the nesting of constraints is not necessarily accompanied by an according nesting of regions in the partonomy. The example above for rule (5) shows that we could also infer new relations from those we know (*HandleCache* must be *part_of* *SelectCache*).

In principle, we could define an *SCCFG* for a non-partonomial spatial structure although this might make the model cognitively more demanding.

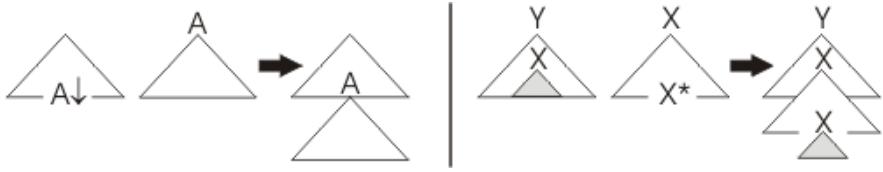


Fig. 4. Substitution (left) and adjoining (right) on a TAG (taken from [20, Fig. 2.2])

3.2 Cross-Dependencies: A Parallel to NLP

Quite frequently, players in CityPoker do not change a playing card although they have found it. They memorize the types of cards they have found and their exact position, and continue in the game. For a number of reasons it might make sense to change in another cache_region first. Sometimes they return to that cache_region at some time later in the game to change a card (without the effort of answering the quiz, cache search, and so on). An intelligent assistance system should recognize the intention *RevisitRegion* and offer an appropriate information service. The *crossed return to region pattern* we would like to model in this use case looks as follows:



What we need for this is a possibility to create *cross-dependencies*. A constrained context-free grammar, like SCCFG, can have cross-dependencies, but only static ones which are defined directly in the rules. No new cross-dependencies can evolve during parsing by the operations offered for CFGs. Modeling all possibilities for cross-dependencies statically in the rules is infeasible, even for CityPoker. Note that more than two constraints might be crossing, and not all *HandleRegion* intentions are followed by an according *RevisitRegion*.

As explained in [7] and [8], similar cross-dependencies occur in NLP. In some natural languages, cross-dependencies are possible between grammatical constructs. If a certain tense, case, or other grammatical form is chosen for the front non- or pre-terminal, we have to choose an according construct for the back non- or pre-terminal. To handle such cross-dependencies, the NLP community has developed formalisms with an extended domain of locality: ‘By a domain of locality we mean the elementary structures of a formalism over which dependencies such as agreement, subcategorization, filler-gap, etc. can be specified.’ ([19, p.5]). In the following, we introduce one of these formalisms, and convert it to a spatially constrained one.

3.3 Tree-Adjoining Grammars

Mildly Context-Sensitive Grammars (MCSG) are a class of formal grammars with common properties [21]. Their expressiveness falls between CFGs and CSGs, and

they support certain kinds of dependencies, including crossed and nested dependencies. They are polynomially parsable and thus especially attractive for mobile intention recognition.

Tree-Adjoining Grammars (TAG), first introduced in [22], are a MCSG with an especially comprehensible way of modeling dependencies. The fundamental difference to CFGs is that TAGs operate on trees, and not on strings. A good introduction to TAG is given by Joshi and Schabes in [20]. They define TAG as follows.

Definition 2. *A Tree-Adjoining Grammar is defined as $TAG = (NT, \Sigma, IT, AT, S)$, where*

- NT are non-terminals
- Σ are terminals.
- IT is a finite set of initial trees. In an initial tree, interior nodes are labeled by non-terminals. The nodes on the frontier (leaf nodes) are labeled by either terminals, or non-terminals. A frontier node labeled with a non-terminal must be marked for substitution. We mark substitution nodes with a \downarrow .
- AT is a finite set of auxiliary trees. In an auxiliary tree, interior nodes are also labeled by non-terminals. Exactly one node at the frontier is the foot node, marked with an asterisk $*$. The foot node must have the same label as the root node. All other frontier nodes are either terminals or substitution nodes, as in the initial trees.
- S is a distinguished non-terminal (starting symbol).

The two operations defined on TAGs are *substitution* and *adjoining* (see Fig. 4). Adjoining is sometimes also called adjunction. Both operations work directly on trees. Substitution is quite straightforward: we can place any initial tree (or any tree that has been derived from an initial tree) headed with a symbol X into a substitution node labeled with $X\downarrow$. It is the adjoining operation that makes TAGs unique: we can adjoin an auxiliary tree labeled with X into an interior node of another tree with the same label. This operation works as follows: (1) we remove the part of the tree which is headed by the interior node, (2) replace it by the auxiliary tree, and (3) attach the partial tree which was removed in step 1 at the foot node. The language defined by a TAG is a set of trees. By traversing a tree we can certainly also interpret it as a String, just like traversing a parse tree of a CFG. If, just for a moment, we try to interpret the two operations as operations on Strings, we see that substitution just replaces a non-terminal by a number of symbols. This is exactly as applying a production rule in a CFG. Adjoining manipulates a String in a more intricate way: a part of the old String (the terminals of the grey tree in Fig. 4) becomes surrounded by new Strings to the left and to the right (by the left and right handside of the X^* in the auxiliary trees).

Joshi and Schabes later add to their definition of TAG the following *Adjoining Constraints*: Selective Adjunction, Null Adjunction, and Obligatory Adjunction. Every non-terminal in any tree may be constrained by one of these. *Selective Adjunction* restrains the auxiliary tree that may be adjoined at that node to a

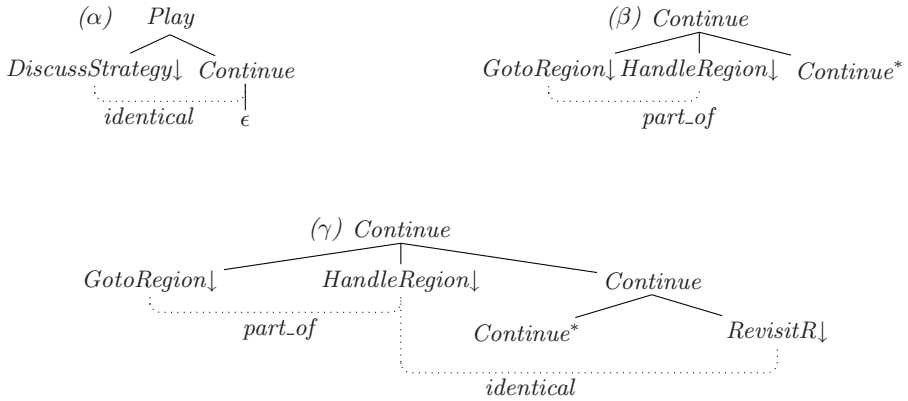


Fig. 5. Initial tree (α) and auxiliary trees (β) and (γ) in a SCTAG for CityPoker

set of auxiliary trees. *Obligatory Adjunction* does the same, but at the same time forces us to do adjoin at that node. *Null Adjunction* disallows any adjunction at that node. These local constraints are important to write sensible grammars, but will not be further discussed here due to our focus on non-local constraints.

A discussion of the formal properties of TAGs, the differences to other grammars, a corresponding automaton, as well as parsing algorithms can be found in a number of publications, e.g. [20,23,24]. For our use case it should be clear that (1) we can easily rewrite any CFG as TAG, (2) TAGs are more expressive than CFGs, and (3) writing a TAG is not necessarily more complicated than writing a CFG. Instead of writing a number of production rules, we just write a number of trees.

3.4 Spatially Constrained Tree-Adjoining Grammars

Definition 3. A *Spatially Constrained Tree-Adjoining Grammar* is defined as $SCTAG = (TAG, R, SR, GC, NLC)$, where

- $TAG = (I, B, IT, AT, S)$, defined over intentions I , and behaviors B .
- R is a set of regions
- SR is a set of spatial relations, where each relation $r \subseteq R \times R$
- $GC \subseteq (IT \cup AT) \times R$ is a set of grounding constraints
- NLC is a set of spatial non-local constraints. Each constraint has a type from the spatial relations SR and is defined for two nodes in one tree from $IT \cup AT$.

This definition applies the idea of spatial constraints to TAGs. The non-local constraints are now defined between nodes in initial/auxiliary trees. The idea of specifying non-local dependencies in TAG is not new. In earlier work on TAGs, Joshi describes this concept as ‘TAGs with links’ [23, Section 6.2].

During the operations of substitution and adjoining the non-local constraints remain in the tree, and become stretched if necessary. Adjoining may also lead to

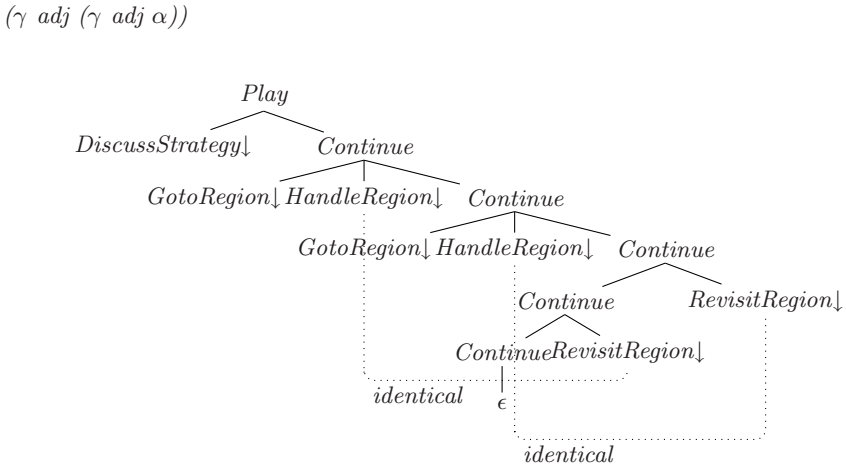
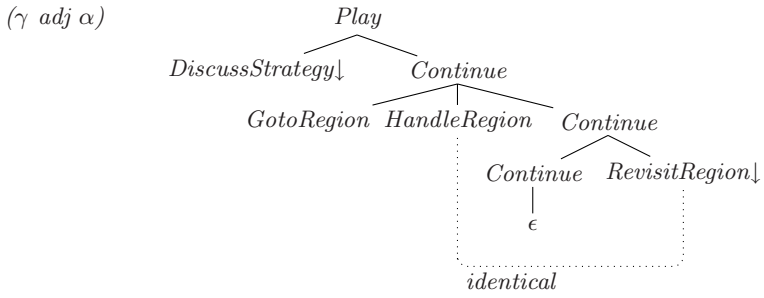


Fig. 6. Adjoining in an SCTAG can lead to cross-dependencies of constraints. Non-crossing spatial constraints are omitted for reasons of clarity.

cross-dependencies like needed for modeling the *crossed return to region pattern*. Figure 5 lists part of a SCTAG that handles the re-visiting of cache_regions in CityPoker. Non-local spatial constraints are displayed as dotted lines. A complete grammar for this use case would convert all context-free rules from Fig. 2 to trees and add them to the grammar. This step is trivial. Figure 6 demonstrates how cross-dependencies evolve through two adjoining operations.

3.5 Parsing Spatially Constrained Grammars

For parsing a spatially constrained grammar, we modify existing parsing algorithms. CFGs are typically handled with chart-based parsers, like the well-known Earley algorithm [25]. An algorithm for parsing TAGs, based on the Cocke-Younger-Kasami algorithm, was proposed in [24], with a polynomial worst and average case complexity. Unfortunately, this complexity is $O(n^6)$ and thus

quite high. Joshi presents a TAG parser that adopts the idea of Earley and improves the average case complexity [20].

We build the parsers for SCCFG and SCTAG on these Earley-like parsers. Earley parsers work on a chart in which the elementary constructs of the grammar are kept, production rules for CFGs, trees for TAGs. A dot in each of these chart entries marks the position up to which this construct has been recognized. In Joshi's parser the 'Earley dot' traverses trees and not Strings. Earley parsers work in three steps: scan, predict, and complete. Predict checks for possible derivations and adds them to a chart. Scan reads the next symbol from the stream and matches it with the chart entries. Complete passes the recognition of rules up the tree until finally we have recognized the starting symbol. The TAG parser has a fourth operation, called 'adjoin', to handle this additional operation.

Our point is that adding spatial constraints to such a parser will not make it slower but faster. The reason is that spatial constraints give us more predictive information. 'Any algorithm should have enough information to know which tokens are to be expected after a given left context' [20, p.36]. Knowing the spatial context of left-hand terminals we can throw away those hypotheses that are not consistent with the spatial constraints. We add this step after each scan operation.

4 Related Work

We started this paper by saying that approaches for intention recognition differ in the way the domain and possible intentions are represented. A number of formalisms has been proposed for modeling the mental state of an agent, ranging from finite state machines [26] to complex cognitive modeling architectures, like the ACT-R architecture [27]. With our formal grammars, which are between these two extremes, we try to keep the balance between expressiveness and computational complexity.

Using formal grammars to describe structural regularities is common, not only in NLP, but also in areas like computer vision [28], and action recognition [29]. Pynadath's state dependent grammars constrain the applicability of a rule dependent on a general state variable [17]. The generality of this state variable leads to an explosion in symbol space if trying to apply a parsing algorithm, so that an inference mechanism is chosen which translates the grammar into a Dynamic Bayes Network (DBN).

Choosing a grammatical approach means using grammars not only for syntax description, but implicitly assigning a certain semantics (in terms of intentions and plans). Linguistics is also concerned with semantics, both, on the sentence level, and on the level of discourse. Webber et al. [30], as one example for the literature on discourse semantics, argue that multiple, possibly overlapping, semantic relations are common in discourse semantics. By using (lexicalized) TAG they describe these relations without the need for building multiple trees.

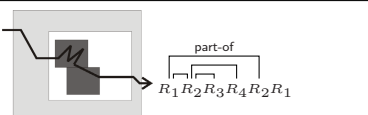
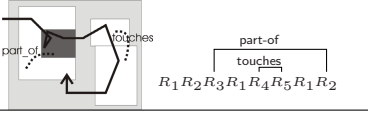

	Dependencies supported	Typical spatial intention pattern	Example
SGIS	Nested: Yes (only part-of relation) Cross: No	Sub-intentions are located in the same or in sub-regions of their parent intention.	
SCCFG	Nested: Yes Cross: No (unless statically defined in productions)	Accidental leaving pattern.	
SCTAG	Nested: Yes Cross: Yes	Crossed return to region pattern.	

Fig. 7. A hierarchy of spatial grammars for mobile intention recognition

Approaches based on probabilistic networks, like DBNs, have widely been applied in plan recognition research. Starting from Charniak and Goldman’s Plan Recognition Bayesian Networks [31], to hierarchical Markov Models as used by Liao et al. in the BELIEVER system [32]. The semantics of ‘goal’ in the latter publication is ‘target location’ without a complex intention model. Bui proposes the Abstract Hidden Markov Memory Model for plan recognition in an intelligent office environment [33]. Geo-referenced DBN are proposed in [34] to fuse sensory data and cope with the problem of inaccurate data.

Intention recognition approaches also differ in the way space is represented: the simplest model consists of a number of points of interest with circular or polygonal areas around them [35,26]. Others add a street network to these locations [32], use spatial tessellation [36], or formalize space with Spatial Conceptual Maps [37].

The quality of our intention recognition relies on a good preprocessing. Converting a motion track into a qualitative representation has been done by a number of researchers, for instance [38]. The authors also compare a number of approaches to generalization. For the classification of segments in Fig. 1 we used a simple decision tree. The set of behavior types we are interested in was chosen manually. An automatic detection of motion patterns is the concern of the spatio-temporal data mining community, see e.g. [39].

One concern of computational and formal linguistics is to find approaches that closely resemble the human conceptualization of language. Steedman, for instance, argues that planned action and natural language are related systems that share the same operations: functional composition and type-raising [40]. Combinatory Categorical Grammar (CCG) is a mildly context-sensitive formalism that supports these operators. Using a ‘spatialized’ version of CCG for mobile intention recognition could be worthwhile. We chose TAG in this paper because we believe that TAG are cognitively more appealing for knowledge engineers not familiar with NLP concepts.

5 Conclusion and Outlook

We have presented a hierarchy of formal grammars for mobile intention recognition: SGIS, SCCFG, and SCTAG. With increasing expressiveness we can handle a larger number of spatio-temporal patterns which frequently occur in scenarios of mobile intention recognition, like in CityPoker. Our grammars allow the knowledge engineer to specify complex intention/space relations by using intuitive spatial relations, instead of writing arbitrarily complex rules for input of behavior/region tuples. Figure 7 gives an overview on the three formalisms.

We only sketched the principle of parsing. Currently, we are specifying the parsing algorithm for SCTAG formally. As a next step we will evaluate the algorithm on the restricted computational resources of a mobile device. In this paper we treated all spatial relations as arbitrary relations, and only mentioned that we could use them for inference. This is also one issue of our future work. Adding temporal constraints could be worthwhile, like ‘the duration between these two intentions may not be longer than a certain Δt ’. Another issues that remains open is recognizing that the agent spontaneously changes her intention [15].

Acknowledgements

I would like to thank Klaus Stein for the discussions on the algorithmic possibilities of SCTAG parsing. Christoph Schlieder’s motivating and constant support of my PhD research made this work possible.

References

1. Schmidt, C., Sridharan, N., Goodson, J.: The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence* 11(1-2), 45–83 (1978)
2. Kautz, H.A.: A Formal Theory of Plan Recognition. PhD thesis, University of Rochester, Rochester, NY (1987)
3. Jarvis, P.A., Lunt, T.F., Myers, K.L.: Identifying terrorist activity with ai plan-recognition technology. *AI Magazine* 26(3), 73–81 (2005)
4. Bui, H.H.: Efficient approximate inference for online probabilistic plan recognition. Technical Report 1/2002, School of Computing Science, Curtin University of Technology, Perth, WA, Australia (2002)
5. Baus, J., Krueger, A., Wahlster, W.: A resource-adaptive mobile navigation system. In: Proc. 7th International Conference on Intelligent User Interfaces, San Francisco, USA, pp. 15–22. ACM Press, New York (2002)
6. Seifert, I., Barkowsky, T., Freksa, C.: Region-Based Representation for Assistance with Spatio-Temporal Planning in Unfamiliar Environments. In: Location Based Services and TeleCartography, pp. 179–192. Springer, Heidelberg (2007)
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), pp. 1612–1617 (2007)

8. Kiefer, P., Schlieder, C.: Exploring context-sensitivity in spatial intention recognition. In: Workshop on Behavior Monitoring and Interpretation, 40th German Conference on Artificial Intelligence (KI-2007), CEUR, vol. 296, pp. 102–116 (2007) 102–116 ISSN 1613-0073
9. Schlieder, C.: Representing the meaning of spatial behavior by spatially grounded intentional systems. In: Rodríguez, M.A., Cruz, I., Levashkin, S., Egenhofer, M.J. (eds.) GeoS 2005. LNCS, vol. 3799, pp. 30–44. Springer, Heidelberg (2005)
10. Egenhofer, M.J., Franzosa, R.D.: Point-set topological relations. *International Journal of Geographical Information Systems* 5(2), 161–174 (1991)
11. Kuhn, W.: Ontologies in support of activities in geographical space. *International Journal of Geographical Information Science* 15(7), 613–631 (2001)
12. Stein, K., Schlieder, C.: Recognition of intentional behavior in spatial partonomies. In: ECAI 2004 Workshop 15: Spatial and Temporal Reasoning (16th European Conference on Artificial Intelligence) (2005)
13. Schlieder, C., Werner, A.: Interpretation of intentional behavior in spatial partonomies. In: Freksa, C., Brauer, W., Habel, C., Wender, K.F. (eds.) *Spatial Cognition III*. LNCS (LNAI), vol. 2685, pp. 401–414. Springer, Heidelberg (2003)
14. Kautz, H., Allen, J.F.: Generalized plan recognition. In: Proc. of the AAAI conference 1986 (1986)
15. Geib, C.W., Goldman, R.P.: Recognizing plan/goal abandonment. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), pp. 1515–1517 (2003)
16. Wooldridge, M.: *Reasoning About Rational Agents*. MIT Press, Cambridge (2000)
17. Pynadath, D.V.: *Probabilistic Grammars for Plan Recognition*. PhD thesis, The University of Michigan (1999)
18. Hirtle, S., Jonides, J.: Evidence of hierarchies in cognitive maps. *Memory and Cognition* 13(3), 208–217 (1985)
19. Joshi, A.K., Vijay-Shanker, K., Weir, D.: The convergence of mildly context-sensitive grammar formalisms. Technical Report MS-CIS-90-01, Department of Computer and Information Science, University of Pennsylvania (1990)
20. Vijay-Shanker, K., Weir, D.: The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory* 27(6), 511–546 (1994)
21. Joshi, A., Levy, L., Takahashi, M.: Tree adjunct grammars. *Journal of Computer and System Sciences* 10, 136–163 (1975)
22. Joshi, A.K., Schabes, Y.: Tree-adjoining grammars. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages*, vol. 3, pp. 69–124. Springer, Berlin (1997)
23. 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*, pp. 206–250. Cambridge University Press, Cambridge (1985)
24. Vijay-Shanker, K., Joshi, A.K.: Some computational properties of tree adjoining grammars. In: Meeting of the Association for Computational Linguistics, Chicago, Illinois, pp. 82–83 (1985)
25. Earley, J.: An efficient context-free parsing algorithm. *Communications of the ACM* 13(2), 94–102 (1970)
26. Dee, H., Hogg, D.: Detecting inexplicable behaviour. In: Proceedings of the British Machine Vision Conference, pp. 477–486. The British Machine Vision Association (2004)
27. Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. *Psychological Review* 111(4), 1036–1060 (2004)

28. 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 (November 2004), <ftp://ftp.cc.gatech.edu/pub/gvu/tr/2004/04-29.pdf>
29. Bobick, A., Ivanov, Y.: Action recognition using probabilistic parsing. In: Proc. of the Conference on Computer Vision and Pattern Recognition, pp. 196–202 (1998)
30. Webber, B., Knott, A., Stone, M., Joshi, A.: Discourse relations: A structural and presuppositional account using lexicalised tag. In: Proc. of the 37th. Annual Meeting of the American Association for Computational Linguistics (ACL1999), pp. 41–48 (1999)
31. Charniak, E., Goldman, R.P.: A bayesian model of plan recognition. *Artificial Intelligence* 64(1), 53–79 (1993)
32. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Learning and inferring transportation routines. *Artificial Intelligence* 171(5-6), 311–331 (2007)
33. Bui, H.H.: A general model for online probabilistic plan recognition. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2003)
34. Brandherm, B., Schwartz, T.: Geo referenced dynamic Bayesian networks for user positioning on mobile systems. In: Strang, T., Linnhoff-Popien, C. (eds.) LoCA 2005. LNCS, vol. 3479, pp. 223–234. Springer, Heidelberg (2005)
35. Ashbrook, D., Starner, T.: Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing* 7(5), 275–286 (2003)
36. Gottfried, B., Witte, J.: Representing spatial activities by spatially contextualised motion patterns. In: RoboCup 2007, International Symposium, pp. 329–336. Springer, Heidelberg (2007)
37. Samaan, N., Karmouch, A.: A mobility prediction architecture based on contextual knowledge and spatial conceptual maps. *IEEE Transactions on Mobile Computing* 4(6), 537–551 (2005)
38. Musto, A., Stein, K., Eisenkolb, A., Röfer, T., Brauer, W., Schill, K.: From motion observation to qualitative motion representation. In: Habel, C., Brauer, W., Freksa, C., Wender, K.F. (eds.) *Spatial Cognition 2000*. LNCS (LNAI), vol. 1849, pp. 115–126. Springer, Heidelberg (2000)
39. 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*, pp. 201–215 (2004)
40. Steedman, M.: Plans, affordances, and combinatory grammar. *Linguistics and Philosophy* 25(5-6), 725–753 (2002)