Modelling, Simulation and Estimation of Situation Histories

Daniel McMichael, Geoff Jarrad, Simon Williams and Michael Kennett
CSIRO Information and Communications Technology Centre
Locked Bag 2
Glen Osmond,
South Australia 5064
Australia.
{Daniel.McMichael, Geoff.Jarrad, Simon.Williams,
Michael.Kennett}@csiro.au

Abstract – *Air situation histories are represented by **sequence set trees**. These structures provide the representational power to model a rich variety of situations with only a small number of rules. The model's generative power makes it a candidate for use in Monte-Carlo testing of planning and surveillance regimes in the air defence domain.*

Sequence sets are sets of sequences that can be constructed recursively into trees. Sequences are represented by sequence grammar trees and the sequences are themselves associated by set grammar trees. These structures are able to provide a detailed hierarchial model of the temporal evolution of systems of targets, their force structure and intent. Red and blue force components and their interactions are represented. Bayesian probabilities can be associated with the sequence set trees and their component sequence sets, and this facilitates robust algorithms for simulating situations and inferring them from data. A sequence set parser, based on the pivot table method, is presented.

Keywords: Situation assessment refinement, air picture, sequence set, probabilistic, situation simulation, situation inference, situation analysis.

1 Introduction

A *situation assessment* is an account of the salient features of the state of a system containing a number of components — such as a military battle or a patient's health. Conventionally, a situation assessment is a freeze-frame picture of the system. While such assessments provide valuable information, they neither model the past state, the processes that have generated the current state, nor do they provide a direct method for extrapolation into the future. This paper sets out a theory and corresponding algorithms required to estimate situation models that account for the events within a system over a period of time and predict its future. It is focused on the requirements of military airborne situation assessment, although its use is by no means restricted to this domain.

The extended situation model we propose covers a finite duration. At any instant within its life, the state of the situation can be evaluated — to give a conventional freeze-frame situation. The extended situation provides a hierarchical representation of the assets and their force structures as they evolve over time.

Our aim is threefold: firstly, to create a rich representation for such situations; secondly, to provide a sampling scheme to enable whole situations to be simulated from high-level summary descriptions; and thirdly, to develop algorithms for extracting situation models from estimated track data.

The paper briefly examines relevant previous work, and in Section 2 presents a scenario referred to in the rest of the paper. Section 3 introduces sequence sets and sequence set trees and describes situation objects, the nodes of situation trees. A graphical notation for situation trees is introduced in Section 4 within the context of the scenario. Set and sequence grammars for modelling the internals of sequence sets are described in Section 5. Probabilistic models for sequence set trees and the grammar trees they comprise are developed in Section 6. These are used to create simulations (Section 7) and to extract situations from track data (Section 8).

1.1 Previous work

The term, situation assessment, derives from military parlance, and refers to a succinct summary of the state of affairs needed to take a decision. While *situation theory* [1] was originally created for an entirely different purpose: the modelling of context in semantics derived from natural language, it has been successfully applied in military situation assessment. In situation theory, logic statements are restricted by a situation. So, for example, the statement "Frederick is bald" is restricted by the situation that "Frederick lives at No. 144b, Acacia Gardens, Port Moresby" — the first statement does not apply to all people called Frederick, only those that live at that address! Lambert [2] has applied this idea of situation in his definition of an *event*, which for him is collection of statements about one or more entities that may be anchored to a location and a time. He applies the term scenario to a collection of consistent events that describe a real-world situation. For him, situations do not exist beyond the relationships and identified roles within them. Information about the situation is extracted by using a deductive inference engine to answer queries.

Laskey and Mahoney [3] have regarded situation analysis as primarily a problem of inference under uncertainty, in which an agent is given probabilistic facts that it then combines in a Bayesian network to provide inferences about

unobserved variables. One of the key insights of situation modelling is the need for *agency* — the attachment of knowledge to decision-making agents, rather than to an all-encompassing knowledge base. Incorporation of agency allows the actions of actors within the situation to be anticipated with greater accuracy.

The locality of knowledge can be partially incorporated into Lambert's method, via his *scenario* construct, which only absorbs information available from that set of events that forms the scenario. In our proposed situation model we attach knowledge to individual agents.

2 A Situation Scenario

To identify some of the issues involved in representing situations, in this section we consider a scenario and show how it can be represented in terms of both force and temporal decompositions. The scenario concerns a raid by the Blue Force over a stretch of sea on a costal target located on land controlled by the Red Force.

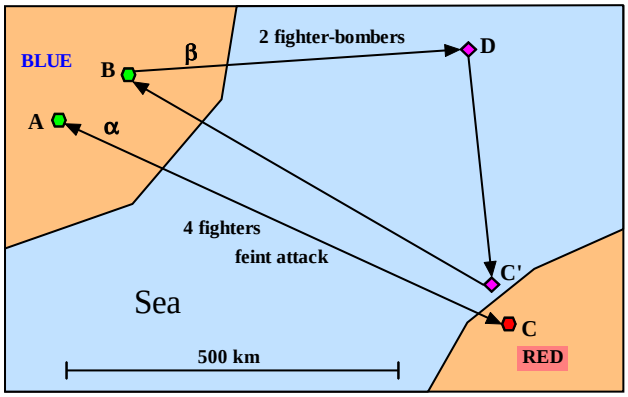

Fig. 1: *A scenario showing a two-formation attack by the blue force on the a red target at C.*

Operation "Sneak Attack". (Fig. 1) The Blue Force attacks the red target over a stretch of water. The Blue Force comprises two formations α and β. Formation α departs from base A, and consists of four fighters that can have ground attack missile capability. However, on this mission, the missiles are not fitted – to increase manoeuvrability and range. This formation operates at high level and carries out a feint attack, intending to draw off enemy defences.

The other formation β consists of two fighter-bombers, and departs base B and heads for the target C indirectly via way-point D. This force pursues a stand-off attack on C with two smart bombs released from one of the aircraft. This force approaches stealthily, flies low, below enemy ground-based radar and is intended only to become visible to the enemy shortly before the attack.

The force and temporal structures of the mission are shown in Figs 2 and 3. Fig. 2 shows the two formations, the aircraft comprising them and the bombs on the attack aircraft. After the attack the bombs are no longer present. The temporal model for the α formation shows it decomposed in two levels; the full lines indicate membership and the dotted lines are *continuity links* that show the temporal sequence.

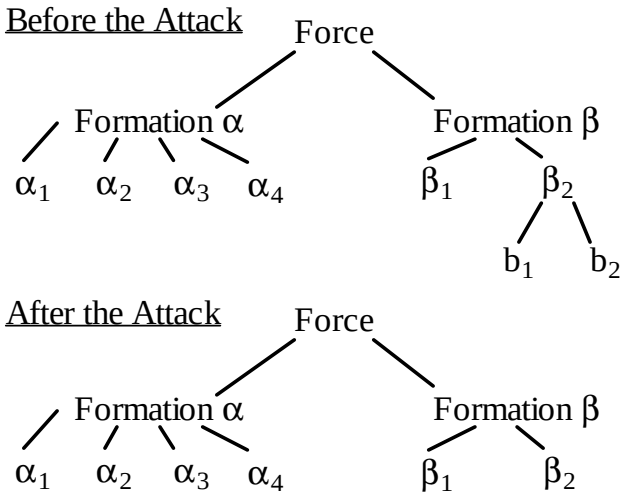

Fig. 2: *Force models for the scenario (above) both before and after the attack. The force comprises two formations α and β that respectively comprise four and two aircraft. One of the aircraft in the β formation is equipped with two smart bombs b_1 and b_2, which are dropped during the attack.*

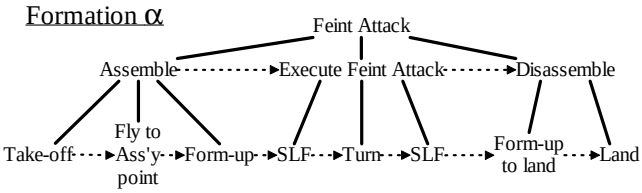

Fig. 3: *Temporal model for the α formation.*

While each of these models is a useful presentation of events, they are incomplete; to represent situations properly a model that jointly represents both force decomposition and temporal structure is required. We now turn to the fundamental mathematical structure required to build such models.

3 Sequence Sets

In this section we present a recurrent model structure based on the *sequence set* primitive, and show how sequence sets can be combined to create *sequence set trees* that can represent a wide class of situations found in air warfare.

In target tracking, an incremental temporal model component is applied recursively to represent the evolution of each track. Analogously, in this approach to situation analysis, a force-temporal model component is applied recurrently to represent situations. It must be capable of recurrence both temporally and force-structurally, so that evolving force structure hierarchies can be represented. The simplest mathematical structure that provides these capabilities is the *sequence set*.

A **sequence set** σ is a set of E sequences. If $e \in \{1, ..., E\}$, then let N_e be the number of stages in the e^{th} sequence. The sequence set also contains a set of Q parameters $\theta = \{\theta_q\}_{q=1}^{Q}$. The sequence of element e is $\bar{\sigma}_e \triangleq \langle \sigma_{ei} \rangle_{i=1}^{N_e}$, A sequence set therefore takes the form:

$$\sigma = \left\{ \left\{ \langle \sigma_{ei} \rangle_{i=1}^{N_e} \right\}_{e=1}^{E}, \theta \right\} = \left\{ \{ \bar{\sigma}_e \}_{e=1}^{E}, \theta \right\}; \quad (1)$$

where the elements σ_{ei} may themselves contain parameters.

Sequence sets can be applied recursively, such that a sequence set can be a member of a higher sequence set. Repeated application of such substitutions generates a *sequence set tree* (Fig. 4). Sequence set trees can represent the evolution of complicated force structures over time.

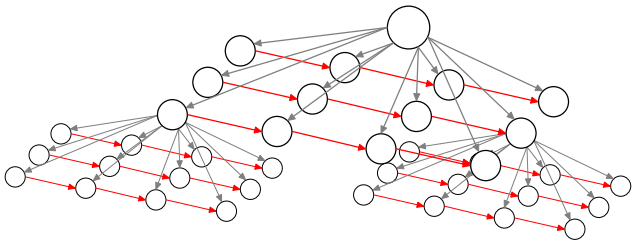

Fig. 4: *A sequence set tree: the root node has three sequences each of four stages; two of its stages are themselves sequence sets.*

Sequence sets can be applied to represent situations by associating each node in the sequence set (a *situation object*) with a set of assets for an interval of time. In our implementation, the lowest level situation objects represent target *trajectory segments*. A trajectory segment is a kinematically coherent section of a target's trajectory history, such as a period of straight level flight (SLF) or a turn. Higher level situation objects represent aggregations of targets, such as multi-aircraft formations (e.g. α and β in Sneak Attack) for longer periods of time. An example of the stages of such an object is the Assemble-Execute-Disassemble temporal sequence of the α-formation's history shown in Fig. 3.

3.1 Situation objects

In the application of sequence set trees to situation modelling, each node of the tree represents the combined activity of the assets that belong to it for its duration. The nodes of the sequence set tree are termed situation objects, and each object includes the following data structures:
Amongst it **state** variables are:

Identifier a unique name for the current object in the situation tree;
Type the current object's type, aircraft, formation, group etc.
Begin and End Time the time span of the object's existence;
Intent friendly/hostile defensive/offensive;
Kinematics a summary of the object's kinematics;
Assets the assets (i.e. aircraft) belonging to it, and their states;
Task a high level description of the goal that the object is attempting to achieve;
Tactic the object's method for carrying out its task.

A set of **links** to its:

Parent The object immediately above the current object in the situation tree;
Predecessors The objects preceding the current object that have **assets** in common with the current object.

4 Situation Trees

The recurrent application of sequence sets, when applied to model a situation, generates *situation trees*, which provide the hierarchical force-temporal situation model we seek. We now examine how the operation Sneak Attack of Section 2 can be represented using a situation tree. To make them visually comprehensible, situation trees are flattened on to two dimensions. In situation tree diagrams, force and temporal structures are expanded in alternate layers down the tree. The rules for constructing these diagrams are:

1. Situation trees comprise nodes for stages ($S, \square$), objects ($O, \bigcirc$) and track segments ($T, \triangle$);
2. The root of a situation tree is a stage node (conventionally at the top of the diagram);
3. Structural dependencies are shown by full links;
4. Structural dependency links can pass down the tree alternately from stage node to object node to stage node, and so on;
5. Continuity links, shown by dashed arrows, are induced when the assets of an object are inherited by a new object;
6. The leaf nodes of the tree are track segments.

A portion of the situation tree for Operation Sneak Attack is shown in Figure 5, which includes both continuity and dependency links. The overall situation is labelled S and below it is the Blue Force object node BF (only Blue Force activity is shown on this diagram). Operation Sneak Attack (SA) is a stage in the activity of the Blue Force and is its child. To execute Operation Sneak Attack, the Blue Force deploys two formations α and β. These are the object child nodes of SA. Formation α is to execute Mission Feint Attack using a number of Blue Force fighter jets flying in formation. Mission Feint Attack consists of three high-level stages: Assemble (As), Execute Feint Attack ($ExFA$) and Disassemble (DAs). Suppose that when generating the situation tree, the Simulator made the stochastic decision that two fighter jets f_1 and f_2 will be used for the mission. In this modified scenario, f_1 and f_2 were based in different locations.

Before the fighter jets can fly in a formation, they must take-off (T) from their respective bases and advance (A) to a meeting point. So to Execute Feint Attack ($ExFA$), α must first execute the Assemble stage (As). This is the first child stage node of α. Whereas α executes As, its members f_1 and f_2 execute T and A separately because they have yet to assemble. When T and A have been completed for both fighter jets, the higher-level stage As is also complete.

The assembly of f_1 and f_2 represent a temporal change in the organisational structure of the situation. Now that the fighter jets are flying together in formation α', they are a single organisation represented in the situation tree by the object node α'. So it is α', and not the individual fighters, that performs the Execute Feint Attack ($ExFA$) stage . $ExFA$ comprises of the stages advance (A) and retreat (R). Since both A and R are basic stages, they cannot be expanded into lower level stages and are directly passed down to f_1 and f_2. Note however, that f_1 and f_2 have slightly different versions of the stages A and R. Different parameters such as the coordinates of the spatial end-point prevent the two fighter jets from travelling the exact same path.

After completing $ExFA$, the fighter jets disassemble (DAs) and return to their respective bases.

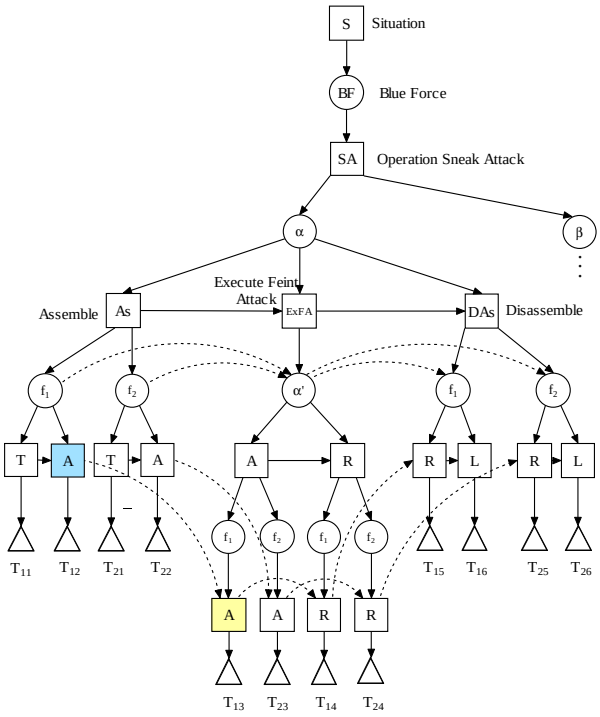

Fig. 5: *A portion of the situation tree for the example introduced in Section 2, and modelled in Section 3. Continuity dependency links are shown.*

5 Grammar for Sequence Sets

We have seen in Sections 2 and 3 that situation trees can be constructed recursively from sequence sets. We now consider the finer grained structures within sequence sets, and show that grammar is an effective tool for modelling both time sequences and force membership.

The expressive power of language follows from its compositionality — it comprises items that can be composed together in a very large number of ways. The grammar of the language is set of syntactical rules for composing word sequences. To provide a similarly versatile model for sequence set generation, we define two categorial grammars [4], one for representing stage sequences and another for aggregating sequences into sequence sets. For the former we define a sequence-grammar and for the latter, a set-grammar. Finally we then show that an entire sequence set can be represented using a unified *sequence set grammar*. The sequence set (1) is refined into a functional form in which grammatical structures replace the simple set operators as follows:

$$\sigma = \mathcal{T}\left(\{\bar{\sigma}_e\}_{e=1}^{E}, \tilde{\theta}\right) \tag{2}$$

$$= \mathcal{T}\left(\left\{\mathcal{S}_e\left(\langle\sigma_{ei}\rangle_{i=1}^{N_e}, \theta_e\right)\right\}_{e=1}^{E}, \tilde{\theta}\right), \tag{3}$$

where $\mathcal{T}\left(\{z_e\}_{e=1}^{E}, \theta\right)$ is the set grammar tree generated with parameters $\theta_{\mathcal{T}}$ over the leaf nodes $\{z_e\}_{e=1}^{E}$. $\mathcal{S}_e\left(\langle\sigma_{ei}\rangle_{i=1}^{N_e}, \theta_e\right)$ is the sequence grammar tree for the e^{th} sequence $\bar{\sigma}_e = \langle\sigma_{ei}\rangle_{i=1}^{N_e}$, generated with parameters θ_e over the leaf nodes $\langle\sigma_{ei}\rangle_{i=1}^{N_e}$. The parameters θ are decomposed as $\theta = \left\{\{\theta_e\}_{e=1}^{E}, \tilde{\theta}\right\}$. We begin by defining the probabilistic *combinatory categorial grammar* (CCG) [5] for encoding the set and sequence grammar trees.

A probabilistic CCG comprises:

- $\mathfrak{N}$, a set of node identifiers $\{n_1, \ldots n_{|\mathfrak{N}|}\}$, each identifying a set of parameters $\mathcal{P}$ which includes a *category* c and the combinator k that formed the node, if present, i.e. $n \rightarrow \mathcal{P};\ \mathcal{P} = \{c, k, \ldots\}$;
- $\mathfrak{K}$, a set of *combinators* $\{k_1, \ldots k_{|\mathfrak{K}|}\}$ defining m-ary rules[1] for combining nodes of the form $(c_1, \ldots, c_m) \Rightarrow_k\ c'$,
- $\mathfrak{R}$, a set of replacement rules $\{r_1, \ldots r_{|\mathfrak{R}|}\}$ defining unary rules on the parameters of the form $\mathcal{P} \Rightarrow_r\ \mathcal{P}'$,
- $\mathfrak{P}_t$, a function assigning a probability to each terminal node $P(\mathcal{P}_n)$, and
- $\mathfrak{P}_n$, a function assigning a probability to each non-terminal node, given its parents: $P(\mathcal{P}_n \mid \mathcal{P}_{n_1}, \ldots, \mathcal{P}_{n_m})$

Applying this definition generates the grammar trees $\mathcal{T}$ and $\mathcal{S}$.

The set grammar comprises the combinators F for functional application, M for modification and $\&$ for conjunction. In function application, one category (the functor) acts on another (the argument) to derive a new category, while in modification the functor merely modifies the argument. Conjunction combines to similar entities. For example, if one node has category $Fo \mid A$, where $Fo =$ 'formation' and $A =$ 'aircraft', the statement that the formation can incorporate an aircraft into itself and yet remain a formation is written $Fo \mid A \cdot A \Rightarrow_F\ Fo$, and can be expressed graphically in the following *set grammar tree*

$$\frac{Fo, F}{Fo \mid A \qquad A}.$$

[1] rules that can accept up to m nodes as input.

The functor category $Fo \mid A$ is on the left; it signifies that under functional application it can absorb an 'A' to produce an Fo. This relationship implies that a formation can be represented by either Fo or $Fo \mid A$.

Sequence grammar trees (parse trees) are similar, but here order is important; so the combinators F and M are prefixed with direction indicators $>$ and $<$. For example, the sequence set with root 'Operation Sneak Attack' in Fig. 5 contains the elements α and β, each with its own sequence. The sequence under the α formation 'Assemble / Execute Feint Attack / Disassemble' can be written as the *sequence grammar tree*:

$$\dfrac{\dfrac{\text{Ass'le, } A \qquad \dfrac{\text{EFA, } Fa{>}F}{\text{EFA, } (Fa/D){<}F}}{\text{EFA, } (Fa/D)\backslash A} \qquad \text{Dis'le, } D}{}\,.$$

where:

Assemble $\rightarrow A$,
Execute Feint Attack (EFA) $\rightarrow (Fa/D)\backslash A$,
Disassemble $\rightarrow D$,

This tree has been annotated with *heads* (e.g. "EFA", "Ass'le") for greater clarity. A set grammar tree combining summary sequence trees for forces α and β to form the sequence set for the operation (marked 'SA' in Fig. 5) yields the following *sequence set grammar tree*:

$$\dfrac{\text{Attack, } At,\ F}{\begin{array}{cc}\text{EFA, } Fa{>}F & \text{Attack, } At \mid Fa \\ \text{etc...} & \text{etc...}\end{array}}\,,$$

where 'etc.' expands into the relevant sequence tree.

Sequence sets constructed in this way provide a rich model for the evolution of situation components. The use of grammar also leads to robust computational mechanisms for extracting situations using parsing algorithms (Section 8).

6 Probabilistic models

Within the framework we have set out, the goal of situation inference is the identification of a credible portfolio of possible situation trees that fit a set of estimated tracks. We pursue a Bayesian approach, in which a probability is associated with a situation tree. That probability is the product of a set of factors, one for each sequence set.

From the definition of our CCG (Section 5), the probability of a grammar tree $\mathcal{G}$ is

$$P(\mathcal{G}) = \prod_{i \in N(\mathcal{G})} P\left(\mathcal{P}_i \mid \{\mathcal{P}_j\}_{j \in A(\mathcal{G},i)}\right), \tag{4}$$

where $\mathcal{G}$ is the grammar tree, $N(\mathcal{G})$ is the set of nodes of tree $\mathcal{G}$, i is an identifier for the nodes in the tree, and $A(\mathcal{G}, i)$ are the direct predecessors of node i on $\mathcal{G}$. A sequence set σ (3) is characterised by the arguments and parameter set

$$\left\{\left\{\langle \sigma_{ei} \rangle_{i=1}^{N_e}\right\}_{e=1}^{E}, \theta\right\}, \tag{5}$$

and the conditional probability of its parameters is

$$P\left(\theta \mid \left\{\langle \theta_{ei} \rangle_{i=1}^{N_e}\right\}_{e=1}^{E}\right) = \prod_{i \in H(\mathcal{T})} P\left(\mathcal{P}_i \mid \{\mathcal{P}_j\}_{j \in A(\mathcal{T},i)}\right) \times \prod_{e \in L(\mathcal{T})} \left\{ \prod_{k \in H(\mathcal{S}_e)} P\left(\mathcal{P}_e \mid \{\mathcal{P}_{ej}\}_{j \in A(\mathcal{S}_e,k)}\right) \times \prod_{i \in L(\mathcal{S}_e)} P\left(\mathcal{P}_{ei} \mid \theta_{ei}\right) \right\}, \tag{6}$$

where $L(\mathcal{T})$ is the set of leaf nodes of $\mathcal{T}$, and $H(\mathcal{T})$ is the set of higher, non-leaf nodes of $\mathcal{T}$. The probability of the entire situation tree Σ is therefore

$$P(\Theta \mid Z) = \prod_{n \in H(\Sigma)} P\left(\theta_n \mid \{\theta_j\}_{j \in A(\sigma_n)}\right) \times \prod_{n \in L(\Sigma)} P(\theta_n \mid z_n), \tag{7}$$

where Θ is set of all the parameters of the tree, z_n is a set of parameters that summarise the n^{th} track segment, and $Z = \{z_n\}_{n \in H(\Sigma)}$. Continuity dependences between nodes hosting assets at consecutive times can be incorporated as additional conditions [6]. While it is mathematically possible to integrate situation extraction with tracking algorithms, this step is not contemplated within the current project.

6.1 Conditional probability factorisation

We now propose a more detailed conditional structure for both set and sequence trees within a sequence set. We present the basis for the factorisation, and then provide candidate factorisations for set and sequence grammar trees in situation assessment.

Let the variables from each of the predecessors of a given node in a grammar tree be assembled into sets of the same type. Let the set of predecessor conditions be y. There are n conditioning variables drawn from each predecessor, and there are m predecessors; so

$$y = \{y_i\}_{i=1}^{n} \quad \text{where} \quad y_i = \{y_{ij}\}_{j=1}^{m}. \tag{8}$$

Let y be covered by an arbitrary set of mutually exclusive subsets, so that $y = \bigcup_j V_j(y)$, and let $\left\{\bar{V}_j(y)\right\}_j$ be an arbitrary conjugate set of subsets subject to

$$V_j(y) \cap \bar{V}_j(y) = \varnothing \ \text{ and } \ V_j(y) \cup \bar{V}_j(y) \subseteq y. \tag{9}$$

Now let the distribution of the conditions given the combinator $P(y \mid k)$ have the independence structure

$$P(y \mid k) = \prod_j P\left(V_j(y) \mid \bar{V}_j(y), k\right). \tag{10}$$

If y' is the set of variables of the derived node corresponding to y, it is possible to show that $P(k, y' \mid y)$ factorises as follows:

$$P(k, y' \mid y) \propto P(k) \prod_j \frac{P\left(k \mid V_j(y), \bar{V}_j(y)\right)}{P\left(k \mid \bar{V}_j(y)\right)} \times \delta\left(V_j(y') - f_j\left(V_j(y'); k\right)\right); \tag{11}$$

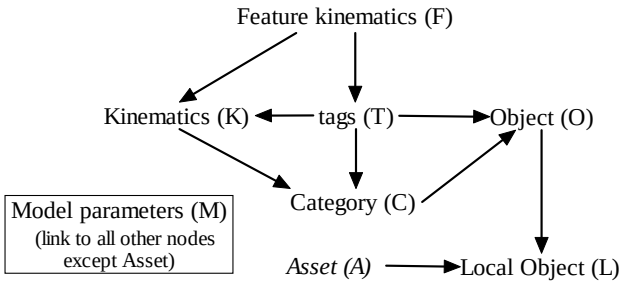

Fig. 6: *Probabilistic dependencies of groups of situation object variables.*

where $f_j(\cdot; k)$ is a function, and $\delta(\cdot) \in \{0, 1\}$ is a delta function.

In situation modelling the variables can be grouped as shown in Fig. 6. The conditional distribution of these variable groups is

$$\begin{aligned} &P(F, T, K, M, C, O, A, k \mid \boldsymbol{F, T, K, M, C, O}, A, Z) \\ &= P(F, k \mid \boldsymbol{F, K, T, M})\, P(T \mid k, \boldsymbol{T, C, K, O, M}) \\ &\times P(C \mid k, \boldsymbol{C, K, T, M})\, P(O \mid \boldsymbol{O, T, C, M})\, P(A \mid Z) \\ &\times P(L \mid \boldsymbol{A, O, M})\, P(K \mid \boldsymbol{K, C, M}), \end{aligned} \tag{12}$$

where bold face indicates variables from the predecessor nodes and light face indicates derived node variables.

The roles of each non-selfexplanatory group are as follows: the *feature kinematics* group hosts filters for detecting such manoeuvres as combat air patrols; the *asset* group provides information extracted directly from the tracks covered by the object; the *model parameters* are house-keeping variables such as *level* and the object's asset list; while the *local object* group fuses information from the object and asset groups.

7 Simulating a Situation

The purpose of a situation simulator is to instantiate a situation: creating all the required nodes between the root and the track segments and generating the tracks.

The simulator comprises a *situation generator* and a *track generator*. The situation generator creates the situation model down as far as track segment level. It provides a script (an elaborate form of flight history) to the track generator, which generates tracks containing such realistic characteristics as dropouts, reappearances, splitting and merging.

7.1 Situation generation

The richness of the situation model structure allows it to generate a very large variety of simulated situations by assembling its components in different ways. The simulator samples the probabilistic model (Section 6) over time and from top to bottom down the situation tree. Sampling accuracy can be improved by first using an approximate top-down model to generate a set of samples and then selecting one by importance sampling of that set using an accurate bottom-up probabilistic model. The situation generator

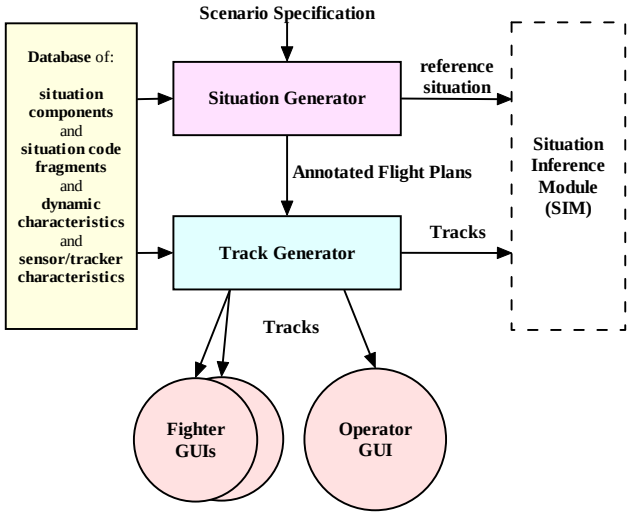

Fig. 7: *The flow of information through the Simulator starting with a situation specification defining the top-level object in the situation, and ending with a collection of track reports transmitted to the GUI's and the Situation Inference Module. The database is a store of interchangeable model fragments needed to flexibly construct the detail of the situation from the specification.*

comprises data, a small tree-generating kernel and a set of code segments that encode tactics that situation objects can implement to generate further objects. A situation object is instantiated by:

1. Copying down the elements of its **state** available from its parent;
2. Preparing to create the next level in the sequence set for which the current node will be the parent:
   (a) Identifying the **tactic** that will be used to achieve the **task**;
   (b) Dividing the **assets** into **teams** that can accomplish the various parts of the **tactic**;
   (c) Creating the **stages** by which each of the **teams** will contribute to the accomplishment of the **tactic**.
3. Instantiating all the child objects that start when it does.

Situation objects are created at their begin times and concluded at their scheduled end times, or earlier, if they are interrupted by events such as a red force incursion, or loss of an asset.

7.2 Track generation

Tracks are simulated by constructing the track state sequences underlying each track segment. Track segments represent object kinematics with cubic splines. Spline parameters are calculated for given endpoints and end times, and when the final time is unknown. Smooth target trajectories are estimated from a list of way points, and for

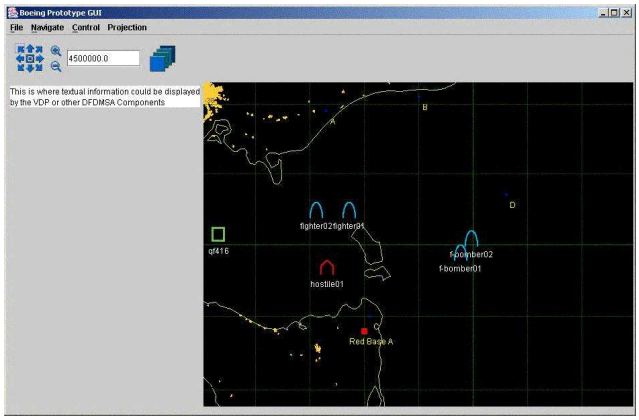

Fig. 8: *A screen shot of the GUI displaying the dynamic simulation of an example scenario at time $t = 2400$. This shows the red force fighter being lured off to the north by a feint attack from the blue force fighters, whilst the blue force bombers approach from the east to attack the red force artillary dump. Also shown for the sake of demonstration is a civilian jet aircraft coming from the west.*

modelling simple target manoeuvres such as climbs, turns and changes of speed.

8 Situation Extraction

This section shows how situation extraction is realised by application of a hierarchical multisequence parsing process we term *sequence set parsing*. Parsing is the estimation of the structure of a grammar tree from partial information about the leaf nodes only. Conventionally, parsing is applied to sequences, but we generalise it to include sets. From there, the construction of sequence set parsing algorithms requires relatively minor additions. For simplicity of presentation, we initially only consider batch evaluation.

A key idea in scalable parsing algorithms is the use of parse tree *forests* to enable the evaluation of the probabilities of large number of parse trees efficiently. A forest is an interwoven set of parse trees, such that all the common nodes are shared. We show that *the same* forest structure can be used in sequence set parsers as in standard sequence parsers.

This section first presents the *pivot table algorithm*, a high-performance sequence parsing algorithm, and then shows how it can be adapted to set parsing. These algorithms can either be used to find the highest probability parses or be detuned to run faster. The trade-off between increased compute speed and decreased parse quality is favourable.

8.1 The pivot table algorithm

Let the sequence of tokens be $\{t_n\}_{n=1}^{N}$. Let a parse tree $\mathcal{S}$ covering tokens t_{n_1} through to t_{n_2} have the properties: begin$(\mathcal{S}) = n_1$ and end$(\mathcal{S}) = n_2$. Let the intertoken position labelled n be the position after token t_n (i.e. the position before the first token is $n = 0$).

In the pivot table algorithm [7] the unary trees provided by the token preprocessor are assembled into a list (the *pivot table*) ordered by probability, with the most probable tree at the bottom. The bottom (most probable) tree is then copied from the list and combinations are formed with the trees available either side of it. These derived trees are inserted into the table such that their probability is lower than or equal to the entry below it and higher than the entry above it. The next tree up is copied out of the list, and the process repeated. The point from which a tree is copied out is known as the *pivot point*. Going up the pivot table, when a tree that spans the entire token sequence is reached, it is necessarily the most probable parse tree.

The reason this algorithm works is that derived trees always have equal or lower probability than the trees that were combined to form them – because their probability is a product of those of their predecessors multiplied by the probability of combination. Thus they are inserted higher up the pivot table than the pivot point. So the first acceptable sequence-spanning terminating tree is reached will have highest probability. This algorithm obtains a parse from the equivalence class of parses with highest probability.

Formal Description. Let the pivot table be $\mathbb{T}$. Let the tree at the bottom of the table be $\mathbb{T}_1$, and the pivot point be p. Left and right tree lists $\{L_n\}_{n=1}^{N}$ and $\{R_n\}_{n=1}^{N}$ are maintained for each intertoken point. The parsing process is exhaustive, and the extracted tree is the one with the highest probability with category belonging to the class of acceptable root categories $\mathcal{R}$, viz:

$\forall\, n \in \{1, \ldots, N\}$ initialise L_n and R_n with ...
  ...the unary trees (track segments)
Put all trees in pivot table $\mathbb{T}$, the most probable lowest ($\mathbb{T}_1$)
Initialise pivot point $p = 1$
While category($\mathbb{T}_p$) $\notin \mathcal{R}$ OR $\mathbb{T}_p$ does not cover ...
  ...the block of input tokens:
  For $(\ell, r) \in L_{\text{begin}(\mathbb{T}_p)} \times \mathbb{T}_p \cup \mathbb{T}_p \times R_{\text{end}(\mathbb{T}_p)}$
    Attempt to combine trees ℓ and r, yielding the set T'
    Merge $\mathbb{T}'$ into $\mathbb{T}$ such that $P(\mathbb{T}_i) \leq P(\mathbb{T}_j) \Leftrightarrow i \leq j$
  $p = p + 1$
To find the best parse:
  backtrack down through the parse tree forest from
    ...root($\mathbb{T}_p$) to extract the entire tree $\mathbb{T}_p$.

8.2 A pivot table algorithm for sets

There is a natural progression from sequence parsing to set parsing via the concept of *combinatory context*. A tree covering a sequence of tokens has a left-combinatory context equal to those trees that end directly before it starts and a right-combinatory context equal to those trees that begin directly after it ends. For sets, there is no notion of order, and so for each tree, there is only one combinatory context, and defining it is a matter of choice. Let the set data at hand be $D = \{d_j\}_{j=1}^{J}$. Let datum d_j have a set of trees $\boldsymbol{T} = \{\mathcal{T}_i\}$ *covering* it (i.e. having d_j as a leaf node). Let the set of trees that can combine with a tree $\mathcal{T}$ be given by the *combinatory context function*, $K(\mathcal{T})$, where the tree $\mathcal{T}$ subsumes its parameters (θ). The *forest of trees* concept also applies in the set parsing domain, as trees can be shared by multiple higher-level trees.

Formal Definition. Let the pivot table be $\mathbb{T}$. Let the tree at the bottom of the table be $\mathbb{T}_1$, and the pivot point be p, and the combinatory context function be $K(\mathbb{T}_i)$. Let there be n set elements to parse. The pivot table set parsing algorithm is set out below.

Initialise pivot table with the unary trees (track segments)
Sort pivot table $\mathbb{T}$, ordered with the most probable lowest (T_1)
Initialise pivot point $p = 1$
While $\mathbb{T}_p \neq$ root or $\mathbb{T}_p$does not cover the input tokens:
  For $(t_1, t_2) \in \mathbb{T}_p \times K(\mathbb{T}_i)$
    Attempt to combine trees $t_1$and t_2, yielding the T'
    Merge $\mathbb{T}'$into $\mathbb{T}$ such that $P(\mathbb{T}_i) \leq P(\mathbb{T}_j) \Leftrightarrow i \leq j$
      ...$P(\mathbb{T}_i) \leq P(\mathbb{T}_j) \Leftrightarrow i \leq j$
  $p = p + 1$
To find the best parse:
  backtrack down through the parse tree forest ...
    ..from root($\mathbb{T}_p$)to extract the entire tree $\mathbb{T}_p$

8.3 Sequence set parsing

Following the structure of the sequence set conditional probability (6) and of the situation tree likelihood (7), situation tree parsing requires the construction of alternate layers of linked sequence and set parse trees from the leaf nodes up (Fig. 9).

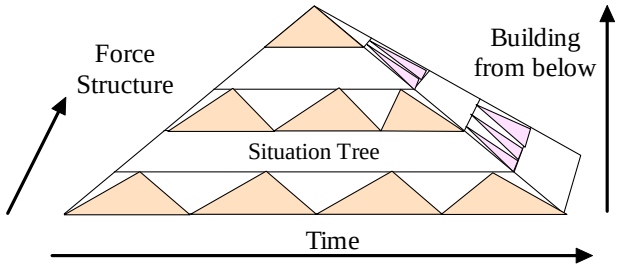

Fig. 9: *A diagramatic view of a situation tree showing alternate layers of set parse trees (determining force structure) and sequence parse trees (determining temporal structure).*

Sequence sets can be parsed using the same tree forest structure as for sets and sequences, because they too are trees. Sequences of track segments from all the tracks are fed to the parser in a batch. Each sequence is identified by its track number $\nu \in \boldsymbol{\nu}$. Each tree in the forest is assigned a level $\ell \in \{1, \ldots, L\}$ that determines with which other trees it can combine. Parsing operates by sharing time between sequence parsing and set parsing at different levels of the sequence set tree forest. The algorithm first parses track segment sequences to form a portfolio of sequence trees. The root node of each tree is assigned a level during the combination process. The algorithm aims to find a set of high probability situation trees that cover all the track segments; it is tabulated below.

Let any situation tree Σ covering all track segments of all tracks satisfy $\Sigma \in C$, and let $\mathbb{P}$ be the portfolio of extracted situation trees covering all the track segments of all the tracks.

Initiate separate sequence pivot tables for the track ...
  ...segments from each track: $\{\mathbb{T}_\nu\}_{\nu \in V}$
Create pivot tables for each level: $S_\ell \ \forall \ell \in \{1, \ldots, L\}$
Initiate S_1 to contain all track segments
Assign all track segments to level 1: $\ell(\mathbb{T}_\nu) = 1 \ \forall \nu \in V$
Repeat K times:
  For $\ell \in \{1, \ldots, L\}$:
    For $\nu \in V$:
      sequence parse T_ν to form $N(\ell)$ new trees
      assign a level ℓ to each new tree
      copy a reference from each new tree in to S_ℓ
    set parse S_ℓ to form $M(\ell)$ new trees
      assign a level ℓ to each new tree
      for each new set of covered tracks:
        create a new sequence pivot table
      add new trees into appropriate T_ν
    For $\Sigma \in S_\ell \cap C$: place reference to C in $\mathbb{P}$
The best situation tree is $\Sigma_{\text{best}} = \arg\max_{\Sigma \in \mathbb{P}} \{P(\Sigma \mid Z)\}$
Extract best tree by backtracking down through the ...
  ...extracted situation tree forest from tree root.

9 Conclusion

The work described in this paper has set out an approach to air situation assessment that is fully consistent with the rigor and clarity of modern tracking algorithms. The representation it describes summarises systems of tracks using discrete grammars to represent higher-level behaviours. Simulation is implemented by sampling, and the estimation process is an extension of existing approaches to parsing.

Acknowledgements

The authors gratefully acknowledge the support of Boeing in this work, particularly of Bob Lobbia and Ken Manus. Very many thanks also to John Colton of CSIRO.

References

[1] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, Cambridge, Masssachusetts, 1983.

[2] Dale A Lambert. Assessing situations. In Robin Evans, Lang White, Daniel McMichael, and Len Sciacca, editors, *Proceedings of Information Decision and Control 99*, pages 503–508, Adelaide, Australia, February 1999. Institute of Electrical and Electronic Engineers, Inc.

[3] K.B. Laskey and S.M. Mahoney. Knowledge and data fusion in probabilistic networks. 2003.

[4] Geoff Jarrad, Simon Williams, and Daniel McMichael. A framework for total parsing. Technical Report 03/10, CSIRO Mathematical and Information Sciences, Adelaide, Australia, January 2003.

[5] M. Steedman. *The Syntactic Process*. MIT Press, 2000.

[6] Daniel McMichael, Geoff Jarrad, Simon Williams, and Mark Cheung. Situation inference I: Extraction. Technical Report 03/08, CSIRO Mathematical and Information Sciences, Adelaide, Australia, January 2003.

[7] T. Matsumoto, D.M.W Powers, and G. Jarrad. Application of search algorithms to natural language processing. In *Proc. Australasian Language Technology Workshop*, Melbourne, December 2003.