Formalizing Battle Management Language: A Grammar for Specifying Orders

Dr. Ulrich Schade
FGAN-FKIE
Neuenahrer Str. 20
Wachtberg, 53343
Germany
0049-228-9435-436
schade@fgan.de

Dr. Michael R. Hieb
Center of Excellence for C4I
George Mason University
4400 University Drive
Fairfax, VA 22030
USA
001-703-993-3990
mhieb@gmu.edu

Keywords: Battle Management Language (BML), Command and Control Information Exchange Data Model (C2IEDM), Command and Control Simulation Interface Language (CCSIL), Command and Control (C2), Grammar, Linguistics, Orders, Reports, Robotics

ABSTRACT: Battle Management Language (BML) is being developed as an open standard that unambiguously specifies Command and Control information, including orders and reports built upon precise representations of tasks. BML is both a methodology and a language specification, based on doctrine and consistent with Coalition standards. Recent work has concentrated on leveraging standard data model semantics (particularly the Command and Control Information Exchange Data Model) for a Simulation Interoperability Standards Organization (SISO) Coalition BML (C-BML) specification. While current BML work has organized task representations around the Command and Control Information Exchange Data Model and the 5 Ws (WHO, WHAT, WHERE, WHEN and WHY), the grammar is implicit rather than explicit.

Development of a formal grammar is necessary for the specification of a complete language. Formalizing BML by defining its grammar should follow the conventions determined by the theory of Linguistics. Initially, it must be determined which type of grammar is to be used. The Chomsky hierarchy specifies that grammars can be Type 0 (unrestricted grammars), Type 1 (context-sensitive grammars), Type 2 (context-free grammars) or Type 3 (regular grammars). While humans typically use a more expressive grammar (Type 1 or Type 2), automated processing is best supported by a more constrained one (Type 2 or Type 3). Our analysis indicates that a Type 2 grammar best fits the requirements for a BML.

To specify a BML grammar, rules must be developed to determine how to create valid BML sentences that describe military tasks. An analysis of US and German Army 5-paragraph orders shows that a pure 5W based grammar can neither cope with all of the expressions needed, nor exclude all sentences that violate our intuition of "correctness". Therefore, rules for BML sentences will require additional and more detailed semantics such that a verb (the 5W's WHAT) determines a structure (expressed as a "frame") for the sentence. This verb frame will then reference the other Ws and additional terms. Rules for the concatenation of BML sentences in our grammar will be guided by NATO STANAG 2014 – "Formats for Orders and Designations of Timings, Locations and Boundaries".

In this paper we propose an initial BML grammar that formalizes the construction of valid BML sentences as well as their concatenation to form military orders. This is illustrated by an example from an Army Order from a Multinational Interoperability Program (MIP) Exercise. We also address the use of this BML grammar in automated systems. Future work includes 1) developing the grammar to express relationships between verbs; 2) further specifying how to concatenate BML sentences; and 3) adding a third class of C2 information to be represented by BML – requests.

1. Need for Formalizing Task Representations in Military Domains

The purpose of this paper is to propose a formalization of Battle Management Language (BML) by developing a *grammar* based upon Linguistic theory. The concept and need for a BML are well documented [1, 7, 10, 23]. To date, BML has defined an approach to resolving ambiguity by leveraging certain existing standards (such as the Command and Control Data Exchange Information Model – the C2IEDM). However, a formal grammar has not been designed, although the need for it has been identified [1].

To be clear about our intent, we view a BML grammar as a subset of a more generic task representation language. We will call this generic language Operational Tasking Language (OTL). While the semantics of a military task have unique aspects, we hypothesize that the syntax is general for a certain class of "Operations" that we define as "a planned activity involving many people performing various actions" [24]. This is similar to the notion of "Action" in the C2IEDM, where an Action is "An activity, or the occurrence of an activity, that may utilize resources and may be focused against an objective." Our approach to a BML grammar, therefore, is to base it on formal Linguistic theory and design it to be applicable to military, peacekeeping, police and fire operations, industrial operations and other general uses. While we realize that the grammar presented here will require review and revision prior to standardization, we hope that this proposal will be a positive contribution to the formalization of BML.

In a general sense, an OTL grammar would be the same as the BML grammar but the specifics of semantics, lexicon and production rules would be different for different domains. Thus an OTL could be specified for disaster relief using a different set of missions and using a different semantics than the C2IEDM.

BML will be useful to the extent that it becomes a standardized "language" that not only has general standards for what should be in an order, but also provides the means for automated systems to distinguish between missions. Currently a human can specify a mission using a C2 system, but the system then only has the name of the mission and some very simple relationships. BML will add "meaning" to the mission by defining parameters that will characterize and distinguish the mission.

To clarify the terms used in the BML grammar we give the following definitions:

Order	A tasking assigned by a superior to a subordinate consisting of one or more tasks
Task	Activity assigned by a superior
Activity	A specific (often skilled) behavior

For completeness, we will briefly introduce the BML concept and give a brief update as to the various organizations involved in defining BML.

1.1 BML Concept

The definition of BML [3] is:

> ***BML is the unambiguous language used to command and control forces and equipment conducting military operations and to provide for situational awareness and a shared, common operational picture.***

The major drawback of using computer-simulated training is the need for large contingents of support personnel to act as workstation controllers and provide the interface between the training unit and the simulation. The group of workstation controllers is often as large as, or larger than, the training audience. While this enables training opportunities at the corps and division echelon, it is still resource-intensive and lacks the degree of fidelity that actual combat operations present to the commander and staff.

Related to this issue of large contingents of workstation controllers, is the lack of effective means to share information and directives between the simulation and the C2 systems. Enabling the C2 systems to not only exchange information but to also allow them to interact directly with the simulation will significantly reduce workstation controller requirements. Good progress has been made in the area of sharing information, however, in the area of controlling the simulation directly from the C2 systems significant progress still needs to be made. This is due to the reliance on unstructured, ambiguous "free text" within the operational C2 messages that are passed within the C2 systems.

"Free text" existing in USMTF, JVMF, and other message formats exists for the benefit of the human. The highly trained, professional soldier has little problem dealing with this "free text." Current automated systems that deal with "free text" handle it as a single data field and pass the <character string> on. Understanding of the content of the <character string> does not exist within the system.

A recent development in simulations is the command agent or intelligent agent software. This type of simulation is designed to receive general "mission type" tasks, and cognitively process the tasks applying a situational awareness. Using this information and by applying knowledge of military doctrine, tactics and techniques it determines its own solution to the problem and then issues appropriate orders and directives to the simulated forces. It subsequently monitors the task's progress against the planned progress. The intelligent agent then makes corrections as necessary. This type of simulation, layered over a more traditional simulation, can greatly reduce the size of the workstation controller contingent. Nevertheless, the introduction of "intelligent agent", "command entities", or other Command Decision Model (CDM) types of software requires unambiguous structures. Free text messages are not an option. A clear, unambiguous Battle Management Language is needed to control these agents.

C2 systems are also evolving. The future systems are incorporating automated decision aids, such as course of action development and analysis tools, and mission rehearsal simulations. While some emerging C2 systems, automatically fill certain fields when operators are entering Operations Orders, this is primarily situational awareness information (e.g. time, location, etc.) and the command information is still carried in free text form.

A predecessor of BML was the Command and Control Simulation Interface Language (CCSIL), a highly structured language for communicating between and among command entities and small units of virtual platforms generated by computers for the Distributed Interactive Simulation (DIS) environment [4]. CCSIL was successful in providing an unambiguous structure, but was not consistent with the emerging C2 data standards and was not maintained as a standard.

1.2 Current Coalition Initiatives

The Simulation to Command and Control Information System Connectivity Experiments (SINCE) program is investigating interoperability issues by conducting multinational C2 experiments, supported by C2 and Simulation systems, designed to address the transformation of collaborative planning and interoperable execution in a coalition environment [13]. This is a US-German Army Bilateral Collaborative Project. SINCE uses a 5W-based Extensible Mark-up Language (XML) schema (as does the current BML concept described in Section 1.3) to represent the various C2 products that embody Information Exchange Requirements (IERs).

Within SISO, the Coalition BML (C-BML) Study Group was formed in September 2004 to investigate the concept of BML and, if warranted, develop a plan to develop a BML Standard. The Study Group has conducted a number of face-to-face and teleconference meetings through the year since the Fall 2004 SIW, involving a membership of over 100 persons from 11 different countries. For more details about the work of the study group see [1]. As the Study Group concluded, it recommended that a Product Development Group (PDG) be formed to standardize the emerging notion of BML. The C-BML Study Group has worked closely with the Military Scenario Definition Language (MSDL) Study Group to coordinate both PDG proposals to ensure a consistent set of standards for initialization, tasking and reporting.

In parallel to the C-BML Study Group activities, the NATO Modeling and Simulation Group (NMSG) established a 12 month Exploratory Team (ET-016) on C-BML [1, 22, 23]. The team, led by France, endorsed the requirement for a C-BML and has proposed that a 3-year Technical Activity Program should be established. Their recommendation was submitted to a meeting of the NMSG in October 2005 in Poland and a NATO Technical Activity (MSG-048) has been approved for 2006-2009.

1.3 Need for a Grammar

To date, the BML initiatives in SISO and NATO have concentrated on using the C2IEDM to disambiguate information pertaining to a task. A set of tables has been identified in the C2IEDM that contains the BML "structure" – the 5Ws. The argument has been advanced that BML is not needed, as the C2IEDM itself is sufficient to represent and disambiguate tasking information. This, we believe, is a shortsighted view. First, although the C2IEDM is a very expressive model that allows an operation to be created, it still needs a standard to represent orders and reports. Second, the C2IEDM is for exchanging facts, but not for communicating meanings and intentions. This, however, is what a language is for.

To be more precise, the missions listed in the C2IEDM (in the "action-task-activity-code" enumerated values) are merely words with a vague textual description. While the C2IEDM is designed to contain all of the information necessary to plan a mission, there is no detailed information on the mission itself. Thus, the "attack" enumeration is never defined using relationships to other objects in the C2IEDM. Or, conversely, the entire context of the mission is described – the weather, the terrain, the control measures that are associated with the overall operation and so on – but the actual mission is never defined beyond a one-word enumeration.

One question that arises is – "If BML is necessary, how can one use the C2IEDM now without it?" The answer is that the current C2IEDM planning implementations rely on human commanders to interpret the definition and assignment of tasks. This is certainly an advancement over previous ways of creating plans and orders, but it limits the use of the C2IEDM by automated systems that do not have skilled commanders available, such as simulations and robots. Furthermore, the lack of a standardized BML (to be used in cooperation with the C2IEDM) will eventually constrain the use of the C2IEDM as more powerful reasoning engines (or "intelligent agents") become available.

A language is used to communicate orders, reports, and requests. The task of the language's grammar is to connect words to communicable expressions. In this sense, it puts together all the necessary information (about a mission and its context) in a way that it can be communicated outside the C2IEDM to a person, to a robot and even to an intelligent agent. The 5Ws are a good start for this purpose.

1.4 Roadmap to Rest of Paper

The remainder of this paper is organized as follows: Section 2 gives a background on the relevant Linguistic theory we will apply to BML. This section will discuss the role a grammar serves a language in general and the role a grammar should serve BML in particular. Section 3 reviews the current BML specifications to determine the scope of an appropriate grammar and presents BML as a context free grammar. Section 4 presents our approach for such a grammar resulting in an initial BML grammar appropriate for general task representation. Section 5 gives an example of using the grammar and Section 6 concludes with recommendations for future research.

2. Development of Formal Grammars

In his book "Syntactic Structures" [5], published in 1957, Noam Chomsky answered the question "What do we know when we know a language?" by postulating that what we know is a set of words (the lexicon of this language) and a set of rules used to generate sequences of those words (sentences of this language). A sequence of words is defined as grammatical if the sequence can be generated by the rules operating on a lexicon.

By this approach, grammaticality does not mean that a sentence is meaningful and thus conveys a message. Chomsky gave the example (1) of a grammatical but not meaningful sequence in order to illustrate this point.

(1) Colorless green ideas sleep furiously.

A formal grammar is defined as an abstract description of a lexicon and rules. It therefore is a precise description of a language; thus a grammar is necessary if one intends to "design" a language like BML that will be processed automatically.

2.1 Applicability of Formal Methods

Following Chomsky's approach, in the field of Linguistics a grammar G is defined as a quadruple, $G = \{S, N, \Sigma, P\}$, where S is the starting symbol, N is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols (the lexicon), and P is a finite set of production rules. A production rule expands a sequence of symbols taken from the union of N and Σ to another sequence of symbols taken from the union of N and Σ. The only restriction is that the left-hand side of a rule must contain at least one non-terminal symbol. The language generated by G, L(G), is the set of all sequences of symbols from Σ which can be produced by applying the rules of P, starting from S. Although N, Σ, and P are finite sets, L(G) need not to be finite because recursion is allowed.

2.2 Types of Grammars

Chomsky defines four types of grammar. They are ordered within what is designated as a Chomsky hierarchy. Grammars of type 0 are unrestricted. Grammars of type 1 have rules of the form $\alpha A \beta \rightarrow \alpha\gamma\beta$ where A is a non-terminal symbol, α, β, and γ are sequences of terminals and non-terminals, and γ consists of at least one symbol. Such a rule can be understood as "A is expanded to γ in the context of α and β". Thus, these kinds of grammars are called *context sensitive* grammars. Grammars of type 2 have rules of the form $A \rightarrow \gamma$ where again A is a non-terminal symbol and γ is a sequence of terminals and non-terminals. Such a rule can be understood as "A is expanded to γ". In contrast to type 1 grammars, no context is to be taken into account. Therefore, these grammars are called *context free* grammars. Grammars of type 3 are even more restricted with respect to their rules. Grammars of type 3 are also called *regular* grammars. Grammars of type 0 and type 3 are not used in practical applications and are not considered further in this paper.

2.3 Syntactic Concepts: Constituency and Subcategorization

In order to state a formal grammar for BML, we have to specify the lexicon (the set of terminal symbols Σ), the set of non-terminal symbols N and the set of production rules P. In order to point out how the specifics of BML reflect in our grammar, we have to introduce some terminology and explain the syntactic concepts *constituency* and *subcategorization*. A complete presentment of the basic concepts of syntax can be found in "Lectures on Contemporary Syntactic Theories" by Peter Sells [19, Chapter 1], a work that also presents and compares some of the main linguistic syntactic theories. Our BML grammar is based on the Lexical Functional Grammar introduced by Kaplan and Bresnan [12] and described more fully in Bresnan [2].

The set of non-terminal symbols can be divided into a set of pre-terminals and a set of constituent symbols. A pre-terminal symbol is a symbol that can be expanded into a terminal symbol or a sequence of terminal symbols. In principle, in order to generate "*move the unit*", the production rule "S → *move the unit*" could be used. Then, S would be a pre-terminal. However, linguists categorize words into classes, traditionally, in verbs, nouns, adjectives, determiners, and so on. This categorization is reflected by production rules like "DET → *the*" or "N → *unit*" representing that *the* is a determiner and *unit* is a noun. V, DET, N and so on are standard pre-terminals.

Considering these word categories, "*move the unit*" can be generated by adding "S → V DET N" to the category rules. However, syntax is more than providing a grammar for the generation of sentences. It also has to assign a meaningful structure to these sentences. Sentences are structured into constituents. With respect to "*move the unit*", "*the unit*" is separated from "*move*". Both are constituents of the sentence, and both can be constituents of other sentences as well, e.g., "*the unit*" is also a constituent of "*resupply the unit*". Constituents can be identified as sequences (of words) answering questions. For example, in the sentence "*advance to phase line Tulip*", "*advance*" answers the WHAT, and "*to phase line Tulip*," answers the WHERE. The idea of the 5W-grammar directly stems from constituency.

Another important syntactic concept is subcategorization. Words do not only belong to a category but sometimes also to a subcategory. This is especially true for verbs. Verbs define what kind of other constituents are allowed or even required in order to form a sentence. For example, "*move*" allows a prepositional phrase specifying a destination like "*towards the assembly area*". In contrast, "*deny*" does not. Subcategorization taps into semantics, especially into the theory of semantic roles [6, 9, 11, 21], but also bears syntactic aspects. With respect to our BML grammar, we will argue in subsections 3.2 and 3.3 that we apply subcategorization to our "verbs". In combination with the Lexical Functional Grammar's principle that syntax is lexically driven we see that in BML a chosen "verb" spans a *frame* that has slots to be filled by constituents. This is further described in 3.2.

3. Design of a BML Grammar

According to the requirements discussed in Section 1, BML is based on the standard data model C2IEDM, since it is concerned with military operations. With respect to a BML grammar this means that the attributes and enumerations provided by the C2IEDM constitute the set of terminal symbols. For example, the C2IEDM table "action-task-activity-code" lists the tasks military units might execute. Therefore, the values given in this table will be verbs in BML. This relationship between BML and the C2IEDM offers the obvious benefit that the definitions the C2IEDM provides for all its attributes and values can be considered as the meanings of these attributes and values. Therefore, the C2IEDM constitutes the lexical semantics for BML. As it is clear that the lexicon (the set of terminal symbols) will be provided by the C2IEDM (according to Chomsky's question "What do we know when we know BML?") we also have to define BML's set of production rules. As a first step, we will restrict this set by defining the type of grammar for BML.

3.1 Analysis of BML requirements to determine the type of Grammar

Determining a grammar for a language means to find the most restrictive grammar (the higher the type the better) that generates the language. Natural languages are supposed to be context-sensitive as proposed by Chomsky [5]. This means that natural languages are supposed to be generated by grammars of type 1. However, BML has to be processed automatically, and the tools (and specific grammars) developed within the field of computational linguistics are restricted to deal with context-free languages, languages generated by grammars of type 2. Therefore, the question is, what do we lose if we give BML a type 2 grammar in order to support automatic processing? (Type 3 grammars do not support constituency; therefore, we do not take them into consideration.) Here is the answer from a classical workbook on computational linguistics: "The fundamental thing that should be kept in mind is that the overwhelming majority of the structures of any natural language can be elegantly and efficiently parsed using context-free parsing techniques" [8, p.133]. With this in mind, we choose BML's grammar to be of type 2.

3.2 Evaluation of 5Ws Concept

In this subsection, we will evaluate the concept of the 5Ws and argue for their evolution into the grammar we are defining. If viewed as a formal language, the 5W concept could define a grammar in which the Ws (WHO, WHAT, WHERE, WHEN and WHY) make up the set of non-terminal symbols. The production rules of such a grammar would have the form W → γ where W is one of the five Ws and γ is a sequence of terminals based on the C2IEDM. Thus, a 5W grammar would be a type 2 grammar as required, and the Ws would be pre-terminals in the terminology given in subsection 2.3. More details of the 5W concept and its mapping into C2IEDM as well as an elaborated example can be found in [10]. This example also illustrates one of the problems of the 5W concept looking from a Linguistic theory viewpoint. In the example ([10] – Figure 7) the WHO is expanded to an organization's name. This organization's relationship to the task (as given by the WHAT) is mapped on C2IEDM's table "organization-action-association". However, this table only expresses relationships like "gives the order for the task" or "observes the task", but not "executes the task". The latter relationship is expressed by "action-resource" in the C2IEDM. Especially with respect to issuing orders, BML must both specify the organization that *orders* a task (the Tasker) and the organization that is ordered to *execute* it (the Taskee). This "split" of the WHO is something we incorporated in our grammar.

There are other problems as well with the implied grammar of the 5Ws. As has been already mentioned, the set of all sequences of terminal symbols that can be generated by applying the rules of a grammar constitutes this grammar's language. These sequences are grammatical sequences. All other sequences are ungrammatical. An ideal grammar would restrict the set of sequences such that a sequence judged as grammatical is a sequence judged as "correct" by an average person and such that a sequence judged as ungrammatical is one judged as "incorrect" by an average person. These judgments are called *intuitions* by linguists, and a grammar based on the 5W concept does not meet our intuitions. Let us consider the examples in (2):

(2a) WHO: 13 (NL) MechBde WHAT: `Rest`
(2b) WHO: 13 (NL) MechBde WHAT: `Support`
(2c) WHO: 13 (NL) MechBde
WHAT: `Rest` 43 (GE) MechBde
(2d) WHO: 13 (NL) MechBde
WHAT: `Support` 43 (GE) MechBde

In all examples above, only WHO and WHAT are given. (2a) is an order to the 13th (NL) Mechanized Brigade to Rest, and (2d) is an order to support the 43rd (GE) Mechanized Brigade, respectively. These orders are correct to our intuitions. However, our intuition judges (2b) – the order to support as incorrect since there is no unit that is supported – and a unit would not support itself. Also, (2c) – the order to Rest the 43rd (GE) Mechanized Brigade – seems incorrect as a unit will "Rest" by being removed from current operations and it is not possible for a unit to perform this for another unit.

Two different kinds of issues can be identified by the analysis of these examples. First, there is the "object problem" which means that a grammar based only on the 5Ws would lack a WHOM. Without a WHOM, task types (the equivalent of a verb) and objectives (the equivalents of verb arguments) cannot be separated, and, therefore, it is necessary to define a huge lexical set of possible WHATs. Indeed, all allowed combinations of action terms like "support" or "rescue" with objective terms like "43 (GE) MechBde" must be inserted into the lexicon as sequences of terminal symbols which might expand the pre-terminal WHAT. This is obviously not practicable. Instead, the grammar should separate the verb from the WHOM-constituent, allowing rules like "WHAT → ***attack*** WHOM" where WHOM is a pre-terminal symbol which can be expanded to the name of any (hostile) unit present in the actual scenario.

The second problem stems from the absence of subcategorization in the 5W implied grammar. Verbs have to be subcategorized. In our view, "frames" should be associated to them such that all verbs spanning a certain frame are members of the same sub-category. A verb's frame defines what can be combined with this verb. For example, in (2) the verb "support" can (and should be) combined with an argument to represent the organization that is supported whereas the verb "Rest" cannot be combined with such an argument.

4. A BML Grammar

In this section, we will present a grammar for describing tasks in the context of an operation for planning and execution. The grammar is designed to specify tasks so that their description can be used in automated systems.

4.1 Scope

The grammar presented in this section is restricted with respect to its scope. The idea behind this is the following. BML has to be developed step by step. Then, in each step, lessons learned during the preceding steps can be applied. We decided to build on the 5Ws concept by developing a "tasking grammar". A tasking grammar is concerned with formalizing orders. At the moment, other kinds of command communication, e.g., reports, are left for future treatment. We decided in favor of orders for two reasons. First, the development of production rules (the set P of a formal grammar) for orders is easier than the development of production rules for reports. Reports include a larger richness of linguistic means, e.g., modality terms like "most probably", "apparently", "possibly" and so on, which are hard to translate into a language written for automatic processing. Second, with respect to C2 systems and simulation systems, the processing of orders is of higher priority than the processing of reports.

The format of orders is defined by the NATO standard STANG 2014 "Format for Orders and Designation of Timings, Locations and Boundaries". An Operational Order is divided into five sections 1) *Situation*, 2) *Mission*, 3) *Execution*, 4) *Administration and Logistics*, 5) *Command and Signal*, and the respective annexes. For conveying the essence of an order to a simulation system, Section 3 is currently the most applicable given the behaviors available. Section 3 will "summarize the overall course of action", "assign specific tasks to each element of the task organization", and "give details of coordination". In the following subsections, we will outline our solution to these aspects.

4.2 Syntax

As has been already said in section 2, a grammar deals with the syntax of a sentence but not with its semantics. This is also true for our tasking grammar. Nevertheless, semantics is an important aspect of a language because in the end content has to be conveyed. So, we will come back to semantics in the next subsection, but start with syntax. In this subsection, we will discuss the production rules of our tasking grammar.

In order to represent the major parts of an order's execution section, our grammar starts with a single rule:

(3) S → B* C_Sp* C_T*

This rule means that the BML order consists of three parts, basic expressions to assign tasks to units, spatial coordination expressions, and temporal coordination expressions. A basic expression is indicated by the non-terminal symbol B, a spatial coordination is indicated by the non-terminal symbol C_Sp, and a temporal coordination by the non-terminal symbol C_T. The star indicates that arbitrarily many of the respective expressions can be stringed together.

In order to avoid the problems we discussed with a grammar based on the 5Ws, the expressions above are composed of a terminal symbol and its frame. To be more precise, a basic expression's terminal symbol is a tasking verb, taken from C2IEDM's table "action-task-activity-code". With respect to basic expressions, the rules have the general form given in (4a). (4b) to (4f) give examples.

(4a) B → Verb Tasker Taskee (Affected|Action) Where Start-When (End-When) Why Label (Mod)*

(4b) B → ***advance*** Tasker Taskee Route-Where Start-When (End-When) Why Label (Mod)*

(4c) B → ***assist*** Tasker Taskee Action At-Where Start-When (End-When) Why Label (Mod)*

(4d) B → ***block*** Tasker Taskee Affected At-Where Start-When (End-When) Why Label (Mod)*

(4e) B → ***defend*** Tasker Taskee Affected At-Where Start-When (End-When) Why Label (Mod)*

(4f) B → ***march*** Tasker Taskee Route-Where Start-When (End-When) Why Label (Mod)*

Tasker is a non-terminal to be expanded by the name of the one who gives the order, Taskee is a non-terminal to be expanded by the name of the unit that is herewith ordered to execute the task, and Start-When and End-When are non-terminals to be expanded by temporal phrases. The temporal phrases for Start-When are given in (5a) and (5b). End-When expands analogously, but is optional as indicated by the brackets. Tasker, Taskee, Start-When, and End-When appear in each basic rule.

(5a) Start-When → **start** Qualifier1 Point_in_Time
(5b) Start-When → **start** Qualifier2 Action

In (5a) and (5b), respectively, Point_in_Time expands to a point in time (a datetime), Action expands to a label which refers to an action, e.g. another task, Qualifier1 expands to a value from C2IEDM's table "action-task-start-qualifier-code", e.g. to **nlt** (not later than), and Qualifier2 expands to a value from table "action-temporal-association-category-code". (5b) refers to a relative point in time, e.g. at the start of a particular action (whenever this may occur).

Affected in (4a), is a non-terminal to be expanded by the name of the one to be affected by the task; in linguistic terms this is the patient. Whether Affected is part of a rule depends on the tasking verb. It is there if the tasking verb's frame requires it as in (4d) and (4e). The same is true for Action in (4a) – separated from Affected by the exclusive or "|" – which occurs in (4c) besides its occurrence in (5b). The same is also true for the Where in (4a). It is either an At-Where or a Route-Where as determined by the verb. A Where has to be expanded by location phrases. These expansions are complex expansions, especially in the case of Route-Where. E.g., Route-Where can be expanded to "**from** Location **to** Location **via** Location **and** Location". Some of the respective phrase rules are given in (6).

(6a) At-Where → **at** Location

(6b) Route-Where → Source Destination Path | Source Path | Destination Path | … | **along** Route

(6c) Source → **from** Location

(6d) Destination → **to** Location

A basic rule ends with the non-terminals Why, Label and the optional Mod. Why represents a reason why the task specified by the rule is ordered. At the moment, it could be expanded by a single tasking verb (a value of "action-task-activity-code"). It is to be seen whether a more complex expansion is necessary, e.g., an expansion by a reduced basic expression. Label is expanded by a unique identifier. By this identifier the single order represented by the respective basic expression can referred to in other expressions, especially in temporal coordinations. The optional Mod (for modifier) is a wild-card that represents additional information necessary to describe a particular task, e.g., formation – to specify a particular formation for an advance, or speed – to specify the speed of a road march.

The abstract rule for spatial coordination is (7a); (7b) and (7c) give examples.

(7a) C_Sp → Control_Feature Tasker (Taskee) Start-When (End-When) Label

(7b) C_Sp →***area of responsibility*** Tasker Taskee Start-When (End-When) Label

(7c) C_Sp → ***hazard area*** Tasker Start-When (End-When) Label

The spatial coordination rules correspond to the basic rules in their form. The key words denote control features, e.g., lines or areas. These are taken from C2IEDM's table "control-feature-type-category-code". In this case the ***area of responsibility*** is assigned by a commander to be used by a subordinate and is considered an area well defined by natural features or control measures for the exclusive operation of the subordinate unit's forces. However, a ***hazard area*** is identified by a unit, but not assigned to a subordinate unit, hence there is no Taskee argument.

The abstract rule for temporal coordination is (8a); (8b) is an example expression, denoting that the action referred to by "label_3_12" is ordered to start exactly when the action referred to by "label_3_11" ends.

(8a) C_T → Temporal-Term Qualifier2 Action Action

(8b) **start at-the-end-of** label_3_12 label_3_11

In temporal coordinations, the non-terminals Action have to be expanded by different unique identifiers that serve as labels for basic expressions. Temporal-Term is either "start" or "end" signifying whether the start or the end of the first Action is determined by the expression. Qualifier2 is expanded by a relational expression that determines how the start (or the end) of the first Action is related to the temporal interval the second Action defines. As has already been said with respect to (5b), Qualifier2 is taken from C2IEDM's table "action-temporal-association-category-code".

Additional examples of BML basic rules and abstract rules are given in Appendix A for a representative sample of C2IEDM tasks and control measures.

4.3 Semantics

As has already been mentioned, the semantics of the terminals are names denoting units and other objects of the real world or are taken from C2IEDM tables. In the latter case, the C2IEDM provides semantic definitions for the terms. The semantic value of the expressions combined from the terminals is in a very concrete sense the action a simulation system executes from it.

5. Example

In order to illustrate how the execution part of an order looks like in BML, we will give an example in this subsection. The original order was used in the "Integrated Operational Test and Evaluation" exercise of the "Multilateral Interoperability Programme (MIP)", September 8th to 26th, in the city of Ede in the Netherlands.

5.1 Example of a Mission Order from the Army Domain

This exercise order is released from the Multi-National Division (West) led by Spain and directed – among others – to the 13th Dutch Mechanized Brigade (M_BDE13(NL)). The following shows some of its content:

3. EXECUTION.

[…]

b) Tasks to Manoeuvre Units.

13 NL MECH BDE:

Phase 1A: Fast Tactical March to PL TULIP by or behind ROUTE DUCK.
Phase 1B: Defense in depth sector EAST, blocking penetration ALFA.
Phase 1C: Assist the rearward passage of the 12 (SP) Cavalry Regiment

In BML this would be translated into

march MND-West(SP) M_BDE13(NL) **along** DUCK **start** **at** Phase1A label_3_11

defend MND-West(SP) M_BDE13(NL) **at** EAST **start** **nlt** Phase1B label_3_12;

block MND-West(SP) M_BDE13(NL) MIR320(BL) **at** TULIP **start** **nlt** Phase1B label_3_13;

assist MND-West(SP) M_BDE13(NL) label_3_57 **at** EAST **start** **nlt** Phase1C label_3_14;

...

In the BML version of the order, the Tasker is the Multi-National Division West, and the Taskee is the 13th Dutch Mechanized Brigade. This is repeated in all basic expressions. Within the WHERE-phrases, the control features are denoted by their names DUCK, EAST, and TULIP. The Start-When-phrases use the key word **start**, qualifiers from C2IEDM's table "action-task-start-qualifier-code", namely **at** and **nlt** ("not later than"), and names which denotes points in time (Phase1A, Phase1B, Phase1C). The last BML sentence (***assist***) illustrates the use of a label. The ***assist*** task has as its object the rearward passage of the 12th Spanish Cavalry Regiment. Note that the Multi-National Division West ordered both the ***assist*** task and the rearward passage task. The rearward passage task received the label label_3_57, which is used to refer to it.

In order to represent the order's "blocking penetration ALFA" directly, the BML representation of the order has to also include the order's section 1a "SITUATION – Enemy Forces" as well. In the representation of this section, the anticipated move of the MIR320(BL) could have been given a label (corresponding to "penetration ALFA") that then could be used in other BML sentences.

Currently there are no Why terms in this example as they remain to be developed in the grammar.

5.2 Preliminary Findings

In order to run a preliminary test for the grammar presented in section 4, we wrote a simple bottom-up shift-reduce parser in PROLOG. The parser takes basic expressions, checks their grammaticality and transforms them into a feature-value matrix. Feature value matrices are a standard format to represent information in computational linguistics. They allow the representation of incomplete information, can be merged by means of unification [20], and can be easily transformed into XML. (In principle, a XML scheme can be calculated which covers all the expressions our grammar allows, whereas the BML expressions are additionally restricted by lexical means in accordance with the design principles of LFG [2]). The transformation of the BML expressions into feature-value matrices also allows us further processing within our SOKRATES system [15, 17, 18]. In particular, the matrices can be enhanced semantically and the results can then be visualized on a map.

Example (8) shows some of the parser's rules and illustrates the principles behind it:

```
(8a) parse([Word|Restwordlist], Stack, Result):-
        reduce([Word|Stack], Reduce_Result),
        parse(Restwordlist, Reduce_Result, Result).

(8b) reduce1([[type: facility|FM], to|Rest],
        [[destination: [type: facility|FM]]|Rest]).

(8c) reduce1([ [start:[type:datetime|TM]],
                [route:RM],
                [type:unit|U2M],
                [type:unit|U1M],
                advance], Result):-
        unify(…, …, Result).
```

(8a) shows the core rule. The parser processes three lists. First, the list of not yet consumed words, second, a stack, and third a list for the parsing result, the matrix to be built up. By one parsing step, a word is shifted from the word list to the top of the stack. Then, the stack is reduced by applying reduce rules. If reduction is no longer possible, the whole process starts anew. In the end, all words are shifted to the stack. In the case of a grammatical input sentence, if all words are consumed and all possible reductions are done, the stack contains the matrix, and nothing else. Otherwise the analyzed sequence of words is not grammatical. (8b) presents a single reduction step. It says that if the top of the stack is a matrix of type facility followed by the word **to**, then these two items can be combined to a matrix of type destination. (8c) presents – in an abbreviated version – another single reduction step: If the stack consists of the task verb ***advance*** at the bottom of the stack and on top of it matrices for the Tasker, the Taskee, a route, and start time, then these matrices can be merged by unification. The result is a matrix for the advance task. (8c) is the parser's abbreviated equivalent to (4a) representing the frame for ***advance***.

6. Conclusions

In this paper we have presented a grammar for BML. By defining the basic phrase in terms of an activity, special coordination and temporal coordination, we believe we have captured the essence of operations. Thus we hypothesize that the grammar is applicable to more general types of operations and that a more general language for operations is possible (as with OTL defined in Section 1).

Of particular interest is the applicability of the grammar to the C-BML prototyping and evaluation. Any BML grammar must be developed and refined through a variety of uses and applications. The grammar must be used to create specific BML content for different domains (e.g. Army, Navy, Air Force). The content for supporting specific missions will consist of elaborating the semantics and defining production rules that are sufficient and necessary for the missions of interest. And once the BML content is established, it then has the potential to increase the standardization of simulation behaviors.

The next step in the development of a BML grammar is the evaluation of a prototype grammar as used by a simulation system. For this purpose, a mapping from the BML defined by the grammar into the language of a simulation system will need to be performed. Then, military orders will need to be translated into the grammar's format. After that, the order can be automatically transferred into the language of the simulation system, and the execution of simulated units evaluated. The results will then give feedback for refinement of the mapping (between BML and the language of the simulation system) and also for the adjustment of the prototype grammar.

This current initial grammar work has focused on "Orders", but we recognize the need for development focused on the C2 information types of "Reports" and also "Requests".

Another future direction for a BML grammar is in the area of semantics. We plan to investigate an assistant system that checks for semantic consistency after an order has been written in BML. Some of the checks this assistant system could make are "Does the Tasker have command and control authority over the Taskee?", "Does the Taskee have the capability and the necessary equipment to execute the ordered task?", and "Is the route selected in the order clear?" These consistent checks will be based on an ontology for military operations [14, 16, 17].

7. Acknowledgements

Dr. Schade performed this work at FGAN's Research Institute for Communication, Information Processing and Ergonomics in cooperation with Bundeswehr IT office, section A5. Dr. Hieb performed this research under the Center for Excellence in C4I at George Mason University. We thank all of the SISO C-BML and NATO MSG-048 participants for their contributions to this area and acknowledge that this work could not have been performed without the intellectual contributions from these individuals. We thank Eric Nielsen, Martin Kleiner and Scott Lambert for their expert assistance on interpreting Army doctrinal terms. We also acknowledge the very thorough review of Curt Blais.

8. References

1) Blais, C., Hieb, M.R., Galvin, K., "Coalition Battle Management Language (C-BML) Study Group Report," 05F-SIW-041, Fall Simulation Interoperability Workshop 2005, Orlando, FL, September 2005.

2) Bresnan, J., *Lexical-Functional Syntax*. Malden, MA: Blackwell, 2001.

3) Carey, S., Kleiner, M., Hieb, M.R. and Brown, R., "Standardizing Battle Management Language – A Vital Move Towards the Army Transformation," Paper 01F-SIW-067, Fall Simulation Interoperability Workshop, 2001.

4) CCSIL Message Content Definitions, Salisbury, M., "Command and Control Simulation Interface Language (CCSIL): Status Update," Twelfth Workshop on Standards for the Interoperability of Defense Simulations, 1995 (http://ms.ie.org/cfor/diswg9503/diswg9503.pdf)

5) Chomsky, N., *Syntactic Structure*. The Hague: Mouton, 1957.

6) Fillmore, C.J., "The Case for Case," In: Bach, E. & Harms, R.T. (Eds.), *Universals in Linguistic Theory*, New York: Holt, Rinehart and Winston, 1968.

7) Galvin, K., "Does the United Kingdom need a Battlespace Management Language?," Paper 04F-SIW-051, Fall Simulation Interoperability Workshop, September 2004.

8) Gazdar, G. & Mellish, C., *Natural Language Processing in PROLOG: An Introduction to Computational Linguistics*. Wokingham, UK: Addison-Wesley, 1989.

9) Gruber, J.S., *Lexical Structures in Syntax and Semantics*. Amsterdam, NL: North Holland, 1976.

10) Hieb, M.R., Tolk, A., Sudnikovich, W.P., and Pullen, J.M.: "Developing Extensible Battle Management Language to Enable Coalition Interoperability," Paper 04E-SIW-064, European Simulation Interoperability Workshop, June 2004.

11) Jackendoff, R.S., *Semantic Structures*. Cambridge, MA: MIT Press, 1990.

12) Kaplan, R. & Bresnan, J., "Lexical-Functional Grammar: A formal system for grammatical representation," In: Bresnan, J. (Ed.), *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press, 1982.

13) Mayk, I., Klose, D., Chan, A., Mai, M. & Negaran, H., "Technical and Operational Design, Implementation and Execution Results for SINCE Experimentation 1," 10th International Command and Control Research and Technology Symposium, Tysons Corner, VA, June 2005.

14) Schade, U., "Towards an Ontology for Army Battle C2 Systems," In: *Proceedings of the 8th ICCRTS, June 17-19, 2003*. National Defense University, Washington, DC, 2003.

15) Schade, U., "Automatic Report Processing," *Proceedings of the 9th International Command and Control Research and Technology Symposium (ICCRTS)*, Command and Control Research Program (CCRP), Copenhagen, September 2004a.

16) Schade, U., "Towards a higher level of interoperability: Ontology components for command and control systems," In: *Proceedings of the NATO R.T.O. IST-Panel Symposium on Coalition C4ISR Architectures and Information Exchange Capabilities*. Den Haag, 2004b.

17) Schade, U. and Frey, M., "Beyond Information Extraction: The Role of Ontology in Military Report Processing," In: Buchberger, E. (Ed.), KONVENS 2004: Beiträge zur 7. Konferenz zur Verarbeitung natürlicher Sprache (Schriftenreihe der Österreichischen Gesellschaft für Artificial Intelligence, Band 5). (pp 177-180), Vienna, Austria, September 2004c.

18) Schade, U., Frey, M. & Becker, S., "From Reports to Maps," In: Bunt, H., Geertzen, J. & Thijse, E. (Eds.), *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-6)* (pp. 407-409), January 12-14, Tilburg, The Netherlands, 2005.

19) Sells, P., *Lectures on Contemporary Syntactic Theories (= CSLI Lecture Notes 3)*. Stanford, CA: CSLI, 1985.

20) Shieber, S.M., *An Introduction to Unification-Based Approaches to Grammar (CSLI Lecture Notes 4)*. Stanford, CA: CSLI, 1986.

21) Sowa, J.F., *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks and Cole, 2000.

22) Tolk, A., Galvin, K., Hieb, M. R., and Khimeche, L., "Coalition Battle Management Language," Paper 04F-SIW-103, Simulation Interoperability Standards Organization, Fall Simulation Interoperability Workshop, Orlando, FL, September 2004.

23) Tolk, A., Hieb, M. R., Galvin, K., and Khimeche, L., "Merging National Battle Management Language Initiatives for NATO Projects," Paper 12 in Proceedings of the RTA/MSG Conference on "M&S to address NATO's new and existing Military Requirements," RTO-MP-123, Koblenz, Germany, October 2004.

24) WordNet, English Dictionary, http://www.wordreference.com, 2006

Author Biographies

ULRICH SCHADE is a Senior Scientist at the Research Institute for Communication, Information Processing and Ergonomics that is part of FGAN financed by the German MoD and is a Lecturer at the Institute for Communication Research and Phonetics, Bonn University. Dr. Schade received his MA in Mathematics in 1986 and his PhD in Linguistics in 1990 at Bielefeld University (Germany), developing a connectionist model for language production processes. He has written many papers and book articles in the areas of Language Production, Ontology Development, and Cognitive Models.

MICHAEL HIEB is a Research Associate Professor with the Center of Excellence in C4I at George Mason University. Dr. Hieb was the Co-Chair of the SISO C-BML Study Group and also was on the team that developed the initial BML concept for the US Army while with Alion Science and Technology. Dr. Hieb was the Technical Director of the MRCI interface project while at SAIC. He received his PhD in Information Technology at George Mason University in 1996, developing an instructable Modular Semi-Automated Forces agent. He has published over 70 papers in the areas of BML, Simulation Interoperability with Command and Control Systems, and Multistrategy Learning.

Appendix A

Examples of basic rules of type B (basic tasks), without the wild-card Mod, for activities taken from the C2IEDM table "action-task-activity-code" [not a complete listing]

B	→	*advance*	Tasker	Taskee		Route-Where	Start-When	(End-When)	Why	Label
B	→	*ambush*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*arrest(legal)*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*arrest(obstruct)*	Tasker	Taskee	Object	At-Where	Start-When	(End-When)	Why	Label
B	→	*assemble*	Tasker	Taskee	Material	At-Where	Start-When	(End-When)	Why	Label
B	→	*assist*	Tasker	Taskee	Action	At-Where	Start-When	(End-When)	Why	Label
B	→	*attack*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*avoid*	Tasker	Taskee	Action	At-Where	Start-When	(End-When)	Why	Label
B	→	*block*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*breach*	Tasker	Taskee	(Affected)	At-Where	Start-When	(End-When)	Why	Label
B	→	*build-up*	Tasker	Taskee	Material	At-Where	Start-When	(End-When)	Why	Label
B	→	*canalize*	Tasker	Taskee	Affected	Route-Where	Start-When	(End-When)	Why	Label
B	→	*capture*	Tasker	Taskee	Material	At-Where	Start-When	(End-When)	Why	Label
B	→	*clear(land)*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*clear(obstacle)*	Tasker	Taskee	Material	At-Where	Start-When	(End-When)	Why	Label
B	→	*concentrate*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*conduct*	Tasker	Taskee	Affected	Route-Where	Start-When	(End-When)	Why	Label
B	→	*confiscate*	Tasker	Taskee	Material	At-Where	Start-When	(End-When)	Why	Label
B	→	*consolidate*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*constitute*	Tasker	Taskee	Object	At-Where	Start-When	(End-When)	Why	Label
B	→	*contain*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*counter attack*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*cover*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*defeat*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*defend*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*deflect*	Tasker	Taskee	Action	At-Where	Start-When	(End-When)	Why	Label
B	→	*demolish*	Tasker	Taskee	Affected	Route-Where	Start-When	(End-When)	Why	Label
B	→	*deny*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*deploy*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*destroy*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*disengage*	Tasker	Taskee	Action	At-Where	Start-When	(End-When)	Why	Label
B	→	*disrupt*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*distribute*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*escort*	Tasker	Taskee	Affected	Route-Where	Start-When	(End-When)	Why	Label
B	→	*evacuate*	Tasker	Taskee	Object	At-Where	Start-When	(End-When)	Why	Label
B	→	*exploit*	Tasker	Taskee	Action	At-Where	Start-When	(End-When)	Why	Label
B	→	*fix*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*guard*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*harass*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*hide*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*hold defensive*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B	→	*hold defensive*	Tasker	Taskee	Feature	At-Where	Start-When	(End-When)	Why	Label
B	→	*hold offensive*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B	→	*hold offensive*	Tasker	Taskee	Feature	At-Where	Start-When	(End-When)	Why	Label
B	→	*identify*	Tasker	Taskee	Object	At-Where	Start-When	(End-When)	Why	Label
B	→	*illuminate*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B	→	*infiltrate*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*isolate*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*mob up*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B	→	*march*	Tasker	Taskee		Route-Where	Start-When	(End-When)	Why	Label

Appendix A (continued)

B →	*move*	Tasker	Taskee		Route-Where	Start-When	(End-When)	Why	Label
B →	*observe*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B →	*observe*	Tasker	Taskee	Feature	At-Where	Start-When	(End-When)	Why	Label
B →	*occupy*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B →	*occupy*	Tasker	Taskee	Freature	At-Where	Start-When	(End-When)	Why	Label
B →	*patrol*	Tasker	Taskee		Route-Where	Start-When	(End-When)	Why	Label
B →	*penetrate*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B →	*plan*	Tasker	Taskee	Action	At-Where	Start-When	(End-When)	Why	Label
B →	*pursue*	Tasker	Taskee	Affected	Route-Where	Start-When	(End-When)	Why	Label
B →	*reconnaissance*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B →	*recover*	Tasker	Taskee	Object	At-Where	Start-When	(End-When)	Why	Label
B →	*reinforce*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B →	*rest*	Tasker	Taskee		At-Where	Start-When	(End-When)	Why	Label
B →	*screen*	Tasker	Taskee	(Affected)	At-Where	Start-When	(End-When)	Why	Label
B →	*secure*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B →	*secure*	Tasker	Taskee	Feature	At-Where	Start-When	(End-When)	Why	Label
B →	*seize*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B →	*seize*	Tasker	Taskee	Feature	At-Where	Start-When	(End-When)	Why	Label
B →	*set up*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B →	*set up*	Tasker	Taskee	Feature	At-Where	Start-When	(End-When)	Why	Label
B →	*set up*	Tasker	Taskee	Facility	At-Where	Start-When	(End-When)	Why	Label
B →	*support*	Tasker	Taskee	Affected	At-Where	Start-When	(End-When)	Why	Label
B →	*withdraw*	Tasker	Taskee	(Affected)	Route-Where	Start-When	(End-When)	Why	Label

Examples of basic rules of type C_S (spatial co-ordinations) for control measures taken from the C2IEDM table "control-feature-type-category-code" (not a complete listing)

C_Sp → *area of interest*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *area of operations*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *area of responsibility*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *assembly area*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *attack position*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *battle position*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *beachhead*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *bridgehead*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *check point*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *drop zone*	Tasker	Taskee	Start-When	(End-When)	Label
C_Sp → *hazard area*	Tasker		Start-When	(End-When)	Label
C_Sp → *key terrain*	Tasker		Start-When	(End-When)	Label
C_Sp → *air corridor*	Tasker	(Taskee)	Start-When	(End-When)	Label
C_Sp → *bomb area*	Tasker	(Taskee)	Start-When	(End-When)	Label