Expressing Knowledge about Protein Interactions in Attempto Controlled English

Diploma Thesis in Computer Science

submitted by
Tobias Kuhn
Zurich, Switzerland
t.kuhn@gmx.ch
Student ID: 01-711-332

Department of Informatics
University of Zurich, Switzerland
Prof. Dr. Michael Hess

Advisors:
Dr. Norbert E. Fuchs (Zurich, Switzerland)
Prof. Dr. Michael Schroeder (Dresden, Germany)

Date of submission: 5 January 2006

Abstract

Scientists are challenged by an ever-increasing number of scientific papers which makes it hard for them to keep track of the relevant literature. A promising solution of this problem is to provide scientists with computational support accessing and evaluating scientific results. A well-known approach is to use natural language processing (NLP) together with the techniques of text mining and summarizing. In this thesis we suggest another approach: the results of scientific papers are summarized by its authors in a formal language, and these formal summaries are added to the papers. In this way we make the results of scientific papers readable and – to some degree – understandable by computers.

As formal language we use Attempto Controlled English (ACE) that is a controlled natural language [7]. Controlled natural languages combine the natural look and feel of natural languages with the processability of formal languages. This makes them to prime candidates for the formal representation of scientific results. Since they are easy to learn and understand, a researcher has not to spend a lot of time learning a complicated formal language.

To demonstrate the practicality of our approach, we build an ontology for protein interactions in ACE and show how scientific results can be expressed using this ontology.

Acknowledgements

First of all, I would like to thank Norbert E. Fuchs who brought me into the fields of logic and controlled natural languages. He made it possible for me to do my diploma thesis on this interesting topic. His support during the four months was indispensable. Furthermore I thank Kaarel Kaljurand for his help on the parser and his general comments to the thesis.

Next, I would like to thank Michael Schroeder who suggested the cooperation between Zurich and Dresden which allowed me to spend one exciting month in Dresden. A big thank goes to Loic Royer who supported me during the time in Dresden and who was a big help in digging into the field of bioinformatics, which was a completely new field for me.

I have to thank all the people of the Attempto group in Zurich and of the bioinformatics group in Dresden. They made my four months of hard work exciting, pleasant, and fruitful.

Last but not least, I would like to thank Michael Hess who was the responsible professor and made it all possible.

Contents

List of Figures

List of Abbreviations

ACE	Attempto Controlled English
APE	Attempto Parsing Engine
DAG	Directed Acyclic Graph
DL	Description Logics
DRS	Discourse Representation Structure
FOL	First-Order Logic
GO	Gene Ontology
KR	Knowledge Representation
MeSH	Medical Subject Headings
NLP	Natural Language Processing
OLF	Ontology Lexicon Format
OWL	Web Ontology Language
RDF	Resource Description Framework
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Chapter 1

Introduction

In this thesis we examine how the language *Attempto Controlled English* (ACE) can be used to summarize the results of scientific papers. To demonstrate the practicality of our approach, we build an ontology for protein interactions in ACE and we show how it can be used to express scientific results.

This chapter explains the motivation, gives a short introduction into the language ACE and into ontologies, takes a look at the common technologies for knowledge representation, and shows some major problems with them. In chapter 2 we introduce the basic formalisms and tools for using ACE as an ontology language. Chapter 3 defines an ontology architecture and chapter 4 shows the creation of an ontology for protein interactions. Chapter 5 describes the dynamic management of knowledge and shows how the results of scientific papers on protein interactions can be summarized in ACE. Chapter 6, finally, reviews the results and draws the conclusions.

1.1 Motivation

Scientists in general and especially biologists are challenged by the increasing amount of scientific papers. *PubMed* is an indexing service for biomedical articles[1] that shows the huge quantity of literature that scientists have to face. *PubMed* contains at the moment 16 million articles and grows by about 600'000 articles per year.

All these articles are written in natural language. That means that we cannot process them directly by computers. But, facing the quantity of papers, it is clear that we need computational support in order to manage the knowledge contained in these papers. There are basically two ways to solve this problem.

First we can try to extract automatically a formal representation from natural language texts. Such attempts are known as *text mining* or *summarizing* and they build upon *natural language processing* (NLP). These techniques are already widely used in different fields of research and there has been a notable progress in the last years, but we will never be able to extract all the information correctly. There is inherent ambiguity and vagueness in natural language that prevents its perfect processing by computers.

[1] `http://www.pubmed.gov`

The second possibility is to force the authors of scientific papers to express their results in a formal language. This allows us to collect the information automatically without any uncertainty. Authors are responsible for the correct formal representation of their results. This requires a lot from the needed formal language which is used for that purpose. On the one hand it has to be easy to learn and easy to understand, since scientists should not have to waste their time on learning a complicated language. On the other hand it has to be expressive enough to represent even complicated scientific results.

It is clear that the second approach is not applicable for papers that are already written without the formal specification. For this reason it is rather a concept for the future than a solution for the today's problem. In this thesis we explore this second approach.

1.2 Ontologies

The term *ontology* is adopted from philosophy and denotes the study of existence and its basic categories. In computer science the term is used for the formal representation of a certain domain of the real world. We adopt the definitions for the terms *conceptualization* and *ontology* from [10].

Conceptualization. A *conceptualization* describes the basic elements for a formal representation of knowledge: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.

Ontology. An *ontology* is an explicit specification of a conceptualization.

The main goal of an ontology is to provide a *shared understanding* of a certain domain. This shared understanding can serve as basis for the communication between people, for the inter-operability between systems, for the improvement of re-usability and reliability of software systems, and for the specification of software [19]. Furthermore ontologies are an excellent basis for the formal representation of knowledge [10].

In this thesis we will use ontologies for knowledge representation. They will give us the foundation for the formal representation of knowledge about protein interactions.

1.3 Knowledge Representation

Knowledge representation (KR) is a multidisciplinary subject that applies theories and techniques from the fields of logic, ontology (in the philosophical reading), and computation theory [18]. We will take a look at four different technologies of knowledge representation: first-order logic (FOL), Description Logics (DL), Web Ontology Language (OWL), and Attempto Controlled English (ACE). At the end we compare these four technologies and explain the advantages of ACE for representing scientific results.

1.3.1 First-Order Logic

Logic is a very old discipline which originated in philosophy. *First-order logic* (FOL)[2] is by far the most widely used, studied, and implemented version of logic [18]. There exists a lot of theoretical research, e.g. about the decidability and complexity of FOL. Thus FOL is a good basis for any kind of knowledge representation.

The notation of FOL consists of constants, functions, variables, predicates, logical operators, and (existential and universal) quantifiers. The constants, functions, and predicates are mapped to the real world and this mapping is called *interpretation*. Constants, functions, and variables stand for objects of the real world. Predicates state relations among such objects and have a truth-value, which is either *true* or *false*. With logical operators we can connect simple statements to build complex ones. Quantifiers, finally, allow to make statements about single – possibly unknown – objects (existential quantification) or about all objects that exist (universal quantification).

The main advantages of FOL are its expressiveness, its thorough formal foundation, and the huge amount of theoretical results.

1.3.2 Description Logics

Description Logics (DL) are a family of knowledge representation languages [4, 14]. As the name suggests, DL builds upon classical logic (i.e. the semantics of the DL notation are founded in logic).

The basic elements of DL are *individuals*, *concepts*, and *roles*. Individuals stand for single objects, concepts for classes of objects, and roles for binary relations between objects. Figure 1.1 shows the basic elements of DL with their FOL representation and their common notation. We will use these three types of elements later as foundation for an ontology in ACE.

DL Element	**FOL**	**Examples**
individual	constant	$JOHN$, $GERMANY$
concept	unary predicate	$Human$, $Protein$
role	binary predicate	$uses$, $brother\_of$

Figure 1.1: Basic Elements of DL

For DL expressions a variable-free notation is used, that is closely related to the notation of set theory. We give some examples now; for a comprehensive formal definition consult [3]. The concept "a man that is not a manager and has a car" is expressed in DL as

$$Man \sqcap \neg Manager \sqcap \exists has.Car$$

where the constructors $\sqcap$, $\neg$, and $\exists$ are used to build a new concept on the basis of the concepts Man, $Manager$, and Car and the role has.

The ontological knowledge in DL is strictly divided into two sections: the *TBox* and the *ABox*. The TBox contains statements that define the terminology, while the ABox contains statements that make assertions about the world by

[2] sometimes also called *first-order predicate calculus*

using the definitions from the TBox. Terminological statements from the TBox look like

$$\begin{aligned} Human \sqcap Male &\equiv Man \\ Human &\sqsubseteq Mammal \end{aligned}$$

The first statement defines that the intersection of the concepts $Human$ and $Male$ is equivalent to the concept Man. The second statement defines that every individual that belongs to the concept $Human$ belongs as well to the concept $Mammal$.

Assertional statements from the ABox look like

$$\begin{aligned} &Man(JOHN) \\ &brother\_of(JOHN, BILL) \end{aligned}$$

The first statement is a *concept assertions* and declares that the individual $JOHN$ belongs to the concept Man. The second statement is a *role assertion* and declares that the individuals $JOHN$ and $BILL$ participate in the role $brother\_of$.

In the development of DL, the complexity of reasoning has been one of the major issues [6]. There is a trade-off between the expressiveness of a representation language and the difficulty of reasoning in that language. Thus DL makes restrictions on the expressiveness in order to allow efficient reasoning. There are different DL languages with different degrees of complexity, but all of them – at least all of the DL languages considered in common DL literature – are less expressive than first-order logic [5].

In contrast to first-order logic, DL is decidable. That means that we can find an algorithm that decides after finite time whether a certain statement is the logical consequence of a set of axioms, or not.

1.3.3 Web Ontology Language (OWL)

Web Ontology Language (OWL) is a knowledge representation language defined by the *World Wide Web Consortium* (W3C)[3]. OWL is based upon RDF (*Resource Description Framework*) which is itself based upon the well-known markup language XML. OWL provides three increasingly expressive sublanguages [13]:

OWL Lite is the least expressive sublanguage. It provides classification hierarchies and simple constraints.

OWL DL is more expressive than OWL Lite, but less expressive than OWL Full. It provides a considerable degree of expressiveness, while it retains computational completeness (i.e. it is decidable). As the name suggests, the structures of OWL DL correspond to the structures of DL.

OWL Full is the most expressive sublanguage with no guaranties on computational completeness.

[3] http://www.w3.org/2004/OWL/

Since OWL builds upon RDF and XML, it has a powerful but complicated syntax. In order to be able to write OWL statements, one has to spend a lot of time learning XML, RDF, and finally OWL. Furthermore XML is designed to be readable by machines, not humans. Thus for a human it is very hard to read OWL files.

1.3.4 Attempto Controlled English

Attempto Controlled English (ACE) is a controlled natural language, i.e. a subset of natural English with a domain-specific vocabulary and a restricted grammar [7]. ACE looks like English, but due to the restrictions it is a formal language. Sentences in ACE can be translated unambiguously into first-order logic.

ACE provides interpretation rules that define the semantics of ACE sentences. Some ACE sentences would be ambiguous in natural English, but the interpretation rules of ACE allow only one of the possible interpretations.

In order to be able to write sentences in ACE, one has to learn the restrictions on the grammar and on the vocabulary. Thus, like every formal language, ACE has to be learned. However, since it looks like natural English, everyone is able to understand sentences in ACE with almost no training. This is a big advantage over other formal languages.

Every ACE sentence can be translated into a representation of first-order logic. This is done by the *Attempto Parsing Engine* (APE)[4]. APE generates a logical representation of ACE sentences and uses for this representation *Discourse Representation Structures* (DRS) [8]. A DRS is equivalent to an expression in common first-order logic.

ACE is designed to achieve a high degree of expressiveness and thus it has to make concessions on the reasoning part. Since ACE has the expressiveness of first-order logic, it is more expressive than DL. But, as consequence, it is harder to reason in ACE than in DL.

1.3.5 Comparison of KR Technologies

We can now compare the four introduced technologies for knowledge representation: first-order logic, Description Logics, OWL, and Attempto Controlled English. Note that these four technologies are not independent. DL and ACE build upon first-order logic, and OWL is inspired by DL. While FOL and DL focus on the theoretical concepts of knowledge, OWL and ACE concentrate on the implementation and application of knowledge representation. Nevertheless we dare to give a direct comparison between these four technologies.

Figure 1.2 shows how the fact 'everyone who is a manager has a car' is expressed in the four different KR technologies. The OWL representation is the most verbose and – from the human perspective – least readable one. The representations in FOL and DL are more concise, but they are still not understandable for people that are not familiar with formal notations. The ACE representation, in contrast, should be immediately understandable for any English speaking person. It looks perfectly like natural English and thus the reader might not even recognize that it is a formal language.

[4] see [2] and `http://www.ifi.unizh.ch/attempto/tools/cape.html`

FOL:	$\forall X(manager(X) \rightarrow \exists Y(car(Y) \wedge has(X,Y)))$
DL:	$Manager \sqsubseteq \exists has.Car$

OWL:	`<owl:Class rdf:ID="Manager">`
	`  <rdfs:subClassOf>`
	`  <owl:Restriction>`
	`  <owl:onProperty rdf:resource="#has"/>`
	`  <owl:someValuesFrom rdf:resource="#Car"/>`
	`  </owl:Restriction>`
	`  </rdfs:subClassOf>`
	`</owl:Class>`

ACE:	Every manager has a car.

Figure 1.2: Example in FOL, DL, OWL, and ACE

As already mentioned, there is a trade-off between the expressiveness of a language and the difficulty of reasoning in it. A language with a high degree of expressiveness has to make concessions to the reasoning, and vice versa.

DL concentrates on a good reasoning performance, whereas ACE has its focus on the expressiveness. OWL does not commit to the one or the other, but provides three sublanguages that allow to choose the right balance between reasoning and expressiveness.

Figure 1.3 gives an overview over the properties of the four knowledge representation technologies. We can state that controlled natural languages like ACE minimize the gap between machines and humans. A reader is able to understand such languages with almost no training. Writing sentences in a controlled natural language is possible with only little effort, especially if the writer is supported by a writing tool (see section 2.3).

	FOL	DL	OWL	(Lite/DL)	(Full)	ACE
conceptual focus	+	+	–			–
focus on application	–	–	+			+
formal definition	+	+	+			+
expressiveness	+	–		–	+	+
reasoning performance	–	+		+	–	–
decidable	–	+		+	–	–
readable by non-specialists	–	–	–			+
easy to learn	–/+	–/+	–			+
practical usage	+	+	+			–

Figure 1.3: Comparison of FOL, DL, OWL, and ACE

Chapter 2

ACE as Ontology Language

This chapter shows how ACE can be used as an ontology language. Concerning the trade-off between expressiveness and efficient reasoning, we focus on maximal expressiveness and, at least for the time being, disregard reasoning issues. Our goal is to make the results of scientific papers accessible by computers as good as possible. We want to minimize the remaining part of the results that cannot be captured. That means that we need a high degree of expressiveness and thus we will get a lower reasoning performance. We want to be able to catch even complicated results without thinking about reasoning at this point, rather than disregard the complicated results just to get a better reasoning performance. Afterwards we are free to reason over only a subset of the results in order to make it more efficient.

For the ontology in ACE we need some basic elements on which we can build. For that reason we adopt the elements of DL: individuals, concepts, and roles[1]. We call them *ontology elements* and use them as the basic elements of ontologies in ACE.

In a first step we will show how the ontology elements are expressed in ACE. For that reason we need to create a lexicon that defines the words. We introduce a new lexicon format that we call *Ontology Lexicon Format* (OLF) and that allows us to define individuals, concepts, and roles as well as their representations in ACE. In addition, it allows us to specify the hierarchy of concepts and roles, and to define the domain and range for each role. Altogether an OLF lexicon defines the basic structure of the ontology.[2]

For the parsing of an ACE text we use the Attempto parser APE. In order to parse an ontology description, we need a lexicon that is compliant to the *ACE Lexicon Format* [1]. For the translation from the Ontology Lexicon Format into the ACE Lexicon Format we need a new tool which we call *Ontology Lexicon Converter*. It allows us to parse an ACE text on the basis of an OLF lexicon.

The user does not need to interact directly with APE or with the Ontology Lexicon Converter. He is supported by a writing assistance tool that hides the technical details. It helps him to use the right words from the lexicon, to use

[1] Note that we adopt only these three basic elements, but *not* the other parts of DL like the constructors, the syntax of terminology definitions, or the syntax of assertions.

[2] [15] suggests to express terminological structures – like concept-hierarchies, role-hierarchies, domains, and ranges – in a controlled natural language as well. We go the other way and express this information in the lexicon.

them as intended by the ontology, and to write sentences that are compliant to the ACE syntax. The user does not need to know about the ACE Lexicon Format and possibly – this depends on the experience of the user – does not need to see the resulting DRS. Figure 2.1 sketches the data flow between the software components and the user.

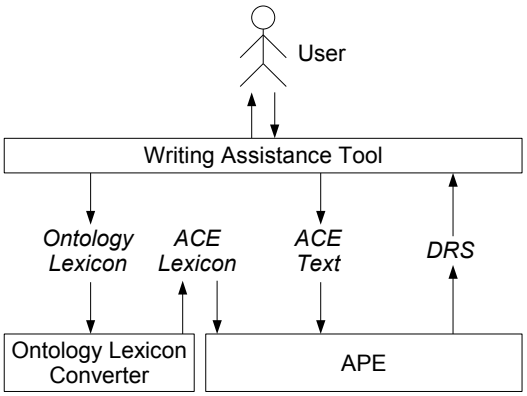


Figure 2.1: Data Flow

In the next sections we describe the Ontology Lexicon Format, the Ontology Lexicon Converter, and we sketch how a writing assistance tool could look like.

2.1 Ontology Lexicon Format

We have now to map the ontology elements – individuals, concepts, and roles – to ACE. For that reason we take first a look at how these elements could be expressed in natural English. In a second step we show how the ontology elements are expressed in ACE and we define the syntax for the entries of the OLF lexicon. After that we will introduce an extended version of roles, which allows us to express context information. Finally we introduce a graphical notation for the basic structure of an ontology in ACE.

2.1.1 Ontology Elements in Natural English

Natural languages are the most common way to describe the real world. In order to find out how we can express the ontology elements in ACE, we take first a look at natural English. Since ACE is a subset of a natural English, we can use the results later for the expression of the ontology elements in ACE.

Individuals in Natural English

Individuals are one of the ontology elements and they represent single entities of the real world. In natural language we can express them with propernames like 'John' or 'Germany'. But sometimes also common nouns are used to express individuals. In the sentence 'everyone knows the king', for example, 'the king'

can stand for an individual. In this case the definite article 'the' is used to indicate that there is just one single entity[3].

Concepts in Natural English

Concepts are the second ontology element. A concept represents a set of entities of the real world that are similar in some way. In natural language we have several possibilities to express such concepts. The most obvious one are common nouns. The noun 'human', for example, could represent the set of all human beings.

But, as a second possibility, we can also use adjectives to represent concepts. Formally spoken they do the same as common nouns: they define a set of entities of the real world. We can express, for example, the set of all men with the adjective 'male'.

Finally we can use intransitive[4] verbs for representing concepts. The set of smokers, for example, can be expressed with the verb 'smokes'.

Roles in Natural English

The third ontology element are roles. They allow us to express binary relations as pairs of individuals. Obviously we can use transitive verbs for that purpose in natural languages. The verb 'knows', for example, can stand for a relationship between two humans.

In natural language we often have sentences like 'a giraffe is an animal from Africa' or 'John is a brother of Bill' where 'is an animal from' or 'is a brother of' represent roles. Such constructs are another important way how roles are represented in natural English.

Furthermore, adverbs are sometimes used to represent roles. If there is a role 'writes', for example, that maps between a human and a text, then the adverb 'manually' might be used to express the role 'writes manually' which is a subrole of 'writes'.

Finally there is the possibility to express a role as a construct with a comparative form of an adjective, like 'is older than'. Such constructs stand typically for roles that represent transitive relationships.

Figure 2.2 shows an overview of how individuals, concepts, and roles can be expressed in natural English.

2.1.2 Ontology Elements in ACE

We show now, how the ontology elements are mapped to ACE and to the DRS, and we specify the syntax of the Ontology Lexicon Format OLF.

An OLF-file is a valid Prolog-file[5], that contains only facts. For these facts only the three predicates `individual`, `concept`, and `role` are used. The detailed syntax of these predicates is shown in the next subsections.

[3] Note that 'the' is also used for other purposes. Not every occurrence of 'the' in natural English denotes an individual.

[4] Intransitive verbs need a subject (like every verb) but no object. For that reason only intransitive verbs are allowed to express concepts.

[5] For this thesis SWI Prolog is used. Consult `http://www.swi-prolog.org/` and [20] for further information.

Element	Natural English
individual	propername; *the* + common noun
concept	common noun; positive form of adjective; intransitive verb
role	transitive verb; transitive verb + adverb; *is a* + noun + preposition; *is* + comparative form of adjective + *than*

Figure 2.2: Ontology Elements in Natural English

Individuals in ACE

In ACE we allow only propernames to stand for individuals. The usage of common nouns with the definite article 'the' for individuals may lead to ambiguity and does not correspond with the ACE interpretation of common nouns. Thus we will not express individuals as common nouns.

Propernames

Propernames that represent individuals in ACE need an OLF lexicon entry that looks like

```
individual(id(ID),
           pn(singular(Singular),
              type(ObjectType),
              gender(Gender))).
```

`ID` is a unique identifier. `Singular` stands for the propername and we have to specify the object type and the gender[6]. We do not allow any form of plural, and thus we can define only singular propernames. As identifier we can simply use the name of the individual, e.g. `'John'` or `'Germany'`.

After the translation into the ACE Lexicon Format (that we will discuss later) we can use the individual as a propername in ACE. Let us take a look at the logical representation (i.e. the DRS). A propername is represented by the following four predicates.

$$object(A, named\_entity, ObjectType)$$
$$quantity(A, cardinality, count\_unit, B, eq, 1)$$
$$structure(A, atomic)$$
$$named(A, Singular)$$

Singular and *ObjectType* stand again for the propername and the object type. The first line declares, that there is an object which is named. This name is indicated in the last line. The two lines in between specify the quantity and the structure of this object. Since we allow only singular forms and no mass

[6] *ObjectType* is one of {unspecified, person, object, time} and *Gender* is one of {human, neutr, masc, fem}. See [1] for further information.

nouns, we use the predicates *quantity* and *structure* only in the form that is introduced above.

Example

Let us have an example how an individual is expressed in OLF. If there is a man called 'John' who we want to express as an individual, then the OLF entry might look like

```
individual(id('John'),
           pn(singular('John'),
              type(person),
              gender(male))).
```

Note that we have to put the name *John* into apostrophes due to the Prolog syntax. Otherwise it would be considered a variable.

Concepts in ACE

To express concepts in ACE we can use all of the three possibilities that we have in natural language: countable common nouns, positive forms of adjectives, and intransitive verbs. We show in this section which lexicon entries we need to create and how the concepts are represented in the DRS. The OLF lexicon entry for concepts looks like

```
concept(id(ID),
        Term,
        superconcepts(SuperconceptList)).
```

where `ID` stands again for a unique identifier that is used for references, `Term` describes the word that is used to represent the concept, and `SuperconceptList` is a list of references to other entries that specify the superconcepts.

As identifier we can again use the name of the concepts, like `human` or `protein`. The list `SuperconceptList` is a list of references to other concepts. The unique identifiers are used for these references. There is no limit on the number of superconcepts; and we can put the empty list `[]`, if we do not want to define a superconcept at all. With such superconcept-definitions we can build hierarchies of concepts.

The exact syntax of the argument `Term` depends on the way how we express the concept. In the following subsections we explain how the different kinds of concepts are expressed in OLF and how they will appear in the DRS.

Countable Common Nouns

The most straightforward way to express concepts in ACE are countable common nouns. In this case the `Term`-Argument for the OLF entry looks like

```
Term = cn(singular(Singular),
          type(ObjectType),
          gender(Gender))
```

where `Singular` is the singular form of the noun that represents our concept. `ObjectType` and `Gender` specify again the object type and gender and again we are not allowed to define a plural form.

Every occurrence of such a countable common noun in an ACE text is represented in the DRS by the three predicates

$$object(A, Singular, ObjectType)$$
$$quantity(A, cardinality, count\_unit, B, eq, 1)$$
$$structure(A, atomic)$$

where *Singular* is again the singular form and *ObjectType* the object type of the corresponding noun. The predicate *object* is the most important one. It introduces an object and assigns it to the specified noun. This object is referenced by the logical variable A. The lines 2 and 3 are exactly the same as for propernames.

Positive Forms of Adjectives

The second possibility to represent a concept in ACE are positive forms of adjectives. In this case the `Term`-argument of the OLF entry looks like

```
Term = adj(positive(Positive))
```

where `Positive` is the positive form of the adjective that represents our concept. Note that we do not specify a comparative and superlative form. We can use this adjective only in its positive form.

Let us again look at the logical representation. If we use an adjective in ACE then it is represented in the DRS as

$$property(A, Positive)$$

where *Positive* is again the positive form of our adjective. The meaning of the predicate *property* is, that the object A has the property that is represented by the adjective *Positive*. Or, from an ontological point of view, the object A belongs to the concept that is represented by *Positive*.

Intransitive Verbs

Finally there are intransitive verbs that are used for representing concepts in ACE. In order to do so, we have to create a lexicon entry that has the argument

```
Term = iv(third_singular(ThirdSingular),
          third_plural(ThirdPlural),
          phrasal_particle(PhrasalParticle))
```

where `ThirdSingular` is the third-singular form of the verb and `PhrasalParticle` is its phrasal particle, if there is any. Since the plural form of verbs is not only used for plural, but also for negation (like 'a man does not smoke'), we need to specify a plural form. Thus `Plural` specifies the plural form, which is only used for negation but not for plural.[7]

An intransitive verb is represented in the DRS as

[7] To be more precise: For the negation in English the 'bare infinitive' of a verb is used, which is homonymous to the plural form.

$predicate(A, unspecified, ThirdSingular, B)$

where $ThirdSingular$ denotes again the verb. It states that the object B participates in the predicate $ThirdSingular$, and the variable A stands for this participation.

Example

Let us again take a look at an example, that shows how to express concepts in OLF. We consider the three concepts 'protein', 'molecule', and 'organic'. We express 'protein' and 'molecule' as common nouns and 'organic' as an adjective. Furthermore the concept 'protein' should be a subconcept of the other two concepts. This could be expressed in OLF as follows.

```
concept(id(molecule),
        cn(singular(molecule),
           type(object),
           gender(neutr)),
        superconcepts([])).

concept(id(organic),
        adj(positive(organic)),
        superconcepts([])).

concept(id(protein),
        cn(singular(protein),
           type(object),
           gender(neutr)),
        superconcepts([molecule, organic])).
```

Roles in ACE

We allow four different possibilities to express roles in ACE. Of the four possibilities that are introduced for natural language, we allow three to be used in ACE without restrictions: transitive verbs, adverbs, and constructs with comparative forms of adjectives. The fourth possibility – constructs with a noun and a predicate – we allow only in a restricted form. The only predicate allowed is 'of' and thus we call them *of*-constructs. This restriction is due to the syntax of ACE. Lexicon entries for roles look like[8]

```
role(id(ID),
     Term,
     superroles(SuperroleList),
     domain(Domain),
     range(Range)).
```

where ID is again a unique identifier. `SuperroleList` defines the superroles, in the same way as it is done for concepts. `Domain` indicates the domain of the role. A domain of a role is a concept that contains all left-side participants of that role. In contrast, `Range` stands for the range of the role, which is the

[8] we will extend this syntax later

concept that contains all right-side participants of the role. If we have a role 'writes', for example, then the domain might be 'person' and the range 'text'. If we do not want to declare a domain or range, we can simply put the empty atom ''.

The declarations of domain and range are inherited through the role hierarchy. That means that such a declaration is not only valid for the corresponding role, but also for all its subroles. Thus the effective domain (or range) of a role is defined by the intersection of all domains (or ranges) that are declared for a – direct or indirect – superrole.

The next subsections show the syntax of the argument `Term` for the four possible ways to express a role.

Transitive Verbs

Transitive verbs seem to be the most straightforward way to express roles. For each role that is expressed as a transitive verb we need a lexicon entry with the argument

```
Term = tv(third_singular(ThirdSingular),
          third_plural(ThirdPlural),
          phrasal_particle(PhrasalParticle),
          direct_preposition(DirectPreposition))
```

where `ThirdSingular` and `ThirdPlural` are the third-singular and third-plural form of the verb. Furthermore we can specify a phrasal particle and a direct preposition. Again the plural form is only needed to express negative statements. In the DRS a transitive verb is represented as follows.

$$predicate(A, unspecified, ThirdSingular, B, C)$$

It states that the objects, denoted by the variables B and C, participate in the predicate that is represented by $ThirdSingular$; and the variable A stands for this participation. It is the same syntax as for intransitive verbs, but with an additional argument.

Adverbs

Let us take a look at the second possibility to express roles in ACE: adverbs. In this case the `Term`-argument of the OLF entry has to look like

```
Term = adv(adverb(Adverb),
           type(ModifierType))
```

where `Adverb` denotes the positive form of the adverb and `ModifierType` stands for its modifier type[9].

Adverbs need to be attached to a transitive verb and thus the logical representation in the DRS looks like

$$predicate(A, state, Verb, B, C)$$
$$modifier(A, ModifierType, none, Adverb)$$

[9] $ModifierType$ is one of {unspecified, manner, time, location, duration, frequency, instrument, destination, comitative}. See [1] for further information.

where *Adverb* and $ModifierType$ are again the positive form of the adverb and the modifier type. $Verb$ stands for the transitive verb that belongs to the adverb. Since an adverb can only be expressed together with a transitive verb, we can use adverbs only if there exists a superrole that is expressed as a transitive verb (e.g. 'writes manually' and 'writes').

Of-Constructs

In order to express a role as an *of*-construct like 'brother of' or 'part of' we need to declare a noun for the OLF entry. This declaration looks exactly the same as for concepts that are expressed as countable common nouns.

```
Term = cn(singular(Singular),
          type(ObjectType),
          gender(Gender))
```

We can use nouns, that are already used for concepts. In this case we do not define the noun, but we just refer to the concept where the noun is defined.

```
Term = cn(ref(RefID))
```

This notation works as well in the other direction, i.e. for concepts that use a noun which is already defined by a role.

The logical representation of an *of*-construct looks like

$$object(A, Singular, ObjectType)$$
$$quantity(A, cardinality, count\_unit, B, eq, 1)$$
$$structure(A, atomic)$$
$$relation(A, Singular, of, C)$$

where $Singular$ stands for the singular form of the noun and $ObjectType$ for its object type. Since we use a noun, the role is represented as an object. The first three lines define this object. The fourth line declares the *of*-relationship. The variables A and C stand for the two objects that participate in our role.

Constructs with Comparative Forms of Adjectives

Last but not least, we can express roles as constructs with comparative forms of adjectives like 'is larger than' or 'is earlier than'. The `Term`-argument has to look like

```
Term = adj(comparative(Comparative))
```

where `Comparative` stands for the comparative form of the adjective. In the DRS such a role is represented as

$$property(A, Comparative, B)$$
$$predicate(C, state, be, D, A)$$

where the variables D and B stand for the objects that participate in our role.

Example

As an example, we pick again the role 'writes' that describes relationships between persons and texts. We assume that there is a superrole which is called 'creates'. The corresponding OLF entry could look like

```
role(id(writes),
     tv(third_singular(writes),
        third_plural(write),
        phrasal_particle(''),
        direct_preposition('')),
     superroles([creates]),
     domain(person),
     range(text)).
```

2.1.3 Context Information for Roles

The examination of the results of scientific papers on protein interactions showed that normal roles are often not sufficient to express the needed information. We can express simple statements like 'P1 interacts-with P2', but we cannot express statements with contextual information like 'P1 interacts-with P2 in Yeast' or 'P1 interacts-with P2 in Microfilament for Motor-Activity'. In order to be able to express such results, we want to allow roles to have such additional information. In natural English we usually express such information with prepositional phrases, and this is exactly the way we will do it in ACE. Figure 2.3 illustrates the examples with context information for roles.

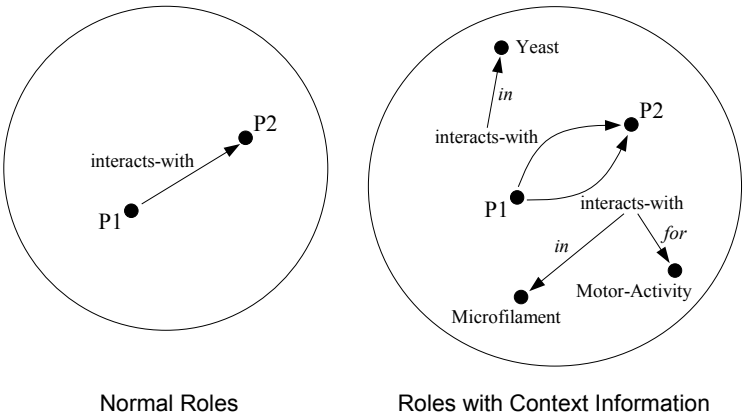


Figure 2.3: Normal Roles and Roles with Context Information

We extend the syntax for role definitions of the Ontology Lexicon Format with an additional argument to specify the allowed prepositions for describing context. The extended syntax looks like

```
role(id(ID),
     Term,
     superroles(SuperroleList),
```

```
        domain(Domain),
        range(Range),
        context(ContextList)).
```

where the list `ContextList` contains pairs of prepositions and corresponding concepts. These concepts define the range of the context-relations for each preposition. The elements of the list look like

```
prep(Preposition, Concept)
```

where `Preposition` stands for the preposition that induces the context, and `Concept` stands for the range of the context-relation. For the role 'interacts-with', as it is shown in the examples above, the `context`-argument might look like

```
context(prep(in, organism),
        prep(in, cellular-component),
        prep(for, molecular-function))
```

where `organism`, `cellular-component`, and `molecular-function` are the IDs of three concepts. The DRS representation of context information looks like

$$predicate(A, unspecified, Verb, B, C)$$
$$modifier(A, unspecified, Preposition, D)$$

where $Verb$ stands for the name of the verb, $Preposition$ for the name of the preposition, and the variable D references the context object. If we do not want to express any context information then we can still use the old version with no `context`-argument.

As for the declaration of domain and range, the declaration of context is inherited through the role hierarchy. That means, if a role declares a preposition for context information then this can be used as well for all its subroles.

2.1.4 The Verb *be*

We have to take a look at the special state of the verb *be*. Generally the verb *be* is handled in ACE in the same way as any other verb, with the only difference that it needs no lexicon entry[10]. In natural English it can be used in three different ways [11]:

Identity. It is used to express identity, e.g. in the sentence 'John is the manager of the pub'.

Predication. Second, it is used for the assignment of properties, like in 'John is unhappy'.

Auxiliary Verb. And third, it is used as an auxiliary verb, e.g. in 'it is raining'.

[10] I.e. it is defined in the built-in lexicon of the parser.

For our ontology in ACE, we use the first two possibilities: identity and predication. In the case of predication, we use the verb *be* for the assignment of concepts and to express roles. Concepts that are represented as nouns or adjectives can be assigned by using the verb *be*. For roles that are represented as *of*-constructs or with comparative forms of adjectives, we need the verb *be* as well.

> John is a man. Every man is male.
>
> John is a customer of Bill. Bill is older than John.

Besides that, we use the verb *be* to express identity. It allows us to express statements as follows.

> A man X is a customer of Bill. A manager Y has a car. The man X is the manager Y.

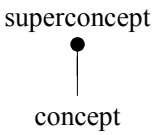
The occurrence of *be* in the first sentence stands for predication, but in the third sentence it stands for identity.

2.1.5 Notation

This section introduces a graphical notation for the basic structure of an ontology as it is expressed in OLF. We show how superconcept- and superrole-relationships are graphically displayed as well as domains and ranges of roles.

Superconcepts

We introduce now a graphical notation for superconcept-relationships. Concepts are displayed by their names in normal font. Superconcept-relationships are indicated by a connection-line with a small filled circle on the side of the superconcept.



If the concept ‘man’, for example, has the superconcept ‘human’, then the graphical notation would look like

Superroles

The graphical notation for superrole-relationships is similar to the one for superconcepts. In contrast to concepts, roles are written in italicized font and the connection-line has a diamond on the side of the superrole.

Let us again have an example. We might have a role 'mother of' that has 'parent of' as its superrole. In this case the graphical representation looks like

Domain and Range of Roles

Finally we show the graphical notation for domain and range of roles. The domain of a role is represented as a simple connection-line between the concept of the domain and the corresponding role. The range is represented as an arrow-style connection-line that points from the role to the concept of the range.

The example we introduced before, where the role 'writes' has the domain 'person' and the range 'text', would look like

2.2 Ontology Lexicon Converter

The Ontology Lexicon Converter translates OLF lexica into the ACE Lexicon Format. This is necessary to use them with APE. If we have, for example, a role 'writes' that we want to express in ACE as a transitive verb then the OLF representation might look like

```
role(id(writes),
     tv(third_singular(writes), third_plural(write),
        phrasal_particle(''), direct_preposition('')),
     superroles([creates]),
     domain(person),
     range(text),
     context([pred(for, journal), pred(with, instrument)])).
```

and after the translation into the ACE Lexicon Format we would get

```
tv(third_singular(writes), third_plural(write),
   third_singular_aliases([]), third_plural_aliases([]),
   type(unspecified),  phrasal_particle(''),
   direct_preposition(''), comment('')).
```

Note that only the information about the verb is used for this translation. The additional information about the superroles, the domain, the range and the context is not considered. But this additional information is important for other tools, like the writing assistance tool (see the next section).

For the Ontology Lexicon Converter there is an implementation in Prolog. Appendix C contains the code.

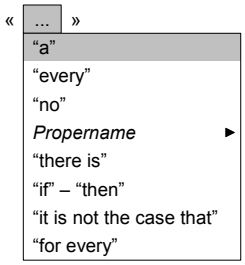
2.3 Writing Assistance Tool

This section sketches a tool that assists to write sentences in ACE using an OLF lexicon. Such a tool allows the user to write specifications in ACE with almost no training. It is similar to the look-ahead editor ECOLE [16].

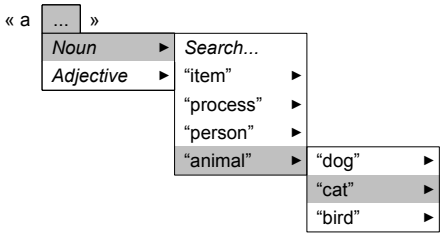
At the beginning there is just an empty sentence that might look like

« ... »

where the quotes indicate the beginning and the end of the sentence and the button in the middle is used to create the content. If the user clicks on it, then a menu is displayed that looks like

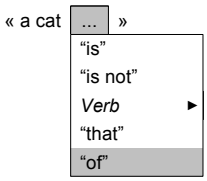


This menu shows the different options how to begin a sentence. The user does not have to know about the syntax of ACE. He sees step by step all the possibilities to continue the sentence. If we choose an entry from this first menu then we get something like

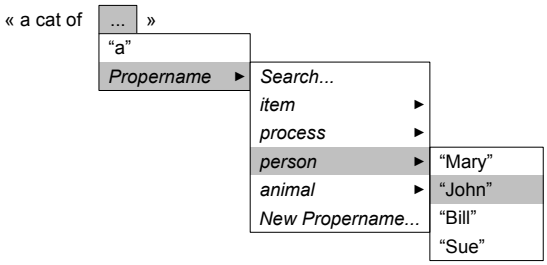


where the article 'a' is now fixed as the beginning of the sentence. Of course there has to be a possibility to undo such decisions, but we will not discuss this here. We have now a new menu with different entries. The menu shows always the possible words at the current position in the sentence, and thus it changes as we proceed.

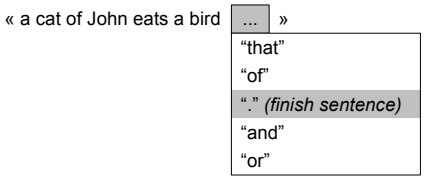
We might want to insert the noun 'cat'. For that purpose we have to navigate through the noun hierarchy. Instead of navigating we could use the search-option which allows us to find an entry, even if we do not know the position in the hierarchy. In the next step we get



where we have the option to specify a verb or to extend the noun phrase with 'that' or 'of'. If we choose 'of' then we get



where we might want to insert a propername. Like common nouns, propernames are structured in hierarchies. After some more steps we might get



where we can finish the sentence.

For the creation of this sentence we did not need any further knowledge about ACE. We might not even have recognized that we used a controlled natural language. Every person that is familiar with English and knows how to handle a simple menu, is able to create sentences in ACE with this tool.

Chapter 3

ACE Ontology Architecture

This chapter introduces an architecture for ontologies in ACE. For that purpose, we use the Ontology Lexicon Format OLF that is introduced in the previous chapter. We could use the same architecture with any another description language.

3.1 Four Tier Architecture

Figure 3.1 shows the basic structure of the ACE ontology architecture. It is a four-tier-architecture with two *terminology tiers* (base terminology and domain terminology) and two *knowledge tiers* (common knowledge and additional knowledge). The distinction of terminology and knowledge is similar to the distinction of a TBox and an ABox in DL.

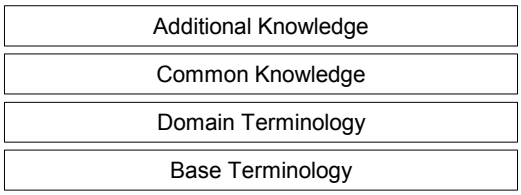


Figure 3.1: The Architecture of an Ontology in ACE

3.1.1 The Terminology Tiers

The two terminology tiers define a terminology for the ontology. A terminology consists of definitions of terms. Furthermore there must exist an interpretation that gives a meaning to the terms. This interpretation is usually expressed in natural language or is implicitly given by the context. It is important to have in mind that such an interpretation has to exist.

We divide the terminology into two tiers: the base terminology and the domain terminology. The base terminology describes fundamental concepts of our world that do not depend on any domain. Terms like '*time*', '*location*',

or '*type*' are useful in every domain, and thus belong to the base terminology. Because there are multiple ways to describe the fundamental concepts of our world, there is more than one possible base terminology.

On the basis of a base terminology we can build domain terminologies. A domain terminology describes the world of a certain domain and thus the terms of the domain terminology are domain-specific. A term can be used in several domain terminologies with several definitions and meanings. 'Complex', for example, may refer to a structure of several proteins in biology, but the same term may refer to a mental disease in psychology.

It is important to notice that there is no knowledge contained in the terminology tiers. They contain just definitions of terms. Such definitions cannot be true or false; they can just be suitable or not for a particular task.

3.1.2 The Knowledge Tiers

The two knowledge tiers declare knowledge about the domain and they use the terms of the underlying terminology tiers.

The common knowledge contains declarations that are widely considered to be true. Since the declarations represent facts of the real world, we can never be completely sure about the truth of these declarations. Thus, both knowledge tiers may contain declarations that are wrong, but the common knowledge should contain only declarations on which the majority of domain experts agree.

Knowledge that is uncertain or not validated belongs to the tier of additional knowledge. This tier gives us the freedom to record declarations that are not (yet) widely accepted. Scientific papers often contain information that belongs to this category.

While the declarations of the common knowledge should always be consistent, this has not to hold for the additional knowledge. We may have several declarations with conflicting information, and we may not know which declarations are true and which are false. Although we are not sure about the truth of such declarations, we want to have them as a part of our ontology.

As we see, the knowledge tiers contain declarations, but not definitions. Declarations may be true or not, and usually we are not able to decide this with certainty.

3.2 Graph Structure of the Ontology

The ontology is structured into four tiers so far. This structure is very coarse and we show in this section how we can structure it on a deeper level. This is necessary for two reasons.

1. We want to be able to build very large ontologies. In order to handle the complexity of such an ontology, we need to create encapsulated modules. Furthermore this gives us the opportunity to improve the performance of reasoning, because we do not have to consider the whole ontology for a certain reasoning task.

2. The terminology and the knowledge of the ontology should be extensible. We want to be able to add entities of terminology or entities of knowledge to an existing ontology. This again requires a modular structure.

We model our ontology as a *directed acyclic graph* (DAG). The nodes of this graph are small modules that represent a part of the terminology or a part of the knowledge. If a module builds upon another module then there is a directed edge from the first to the latter. If a module A, for example, builds upon the modules B and C then there is an edge from A to B and from A to C.

We distinguish two kinds of nodes. *Terminology-nodes* (or *T-nodes*) contain terminological information and belong to one of the terminology tiers. *Knowledge-nodes* (or *K-nodes*) contain knowledge information and belong to one of the knowledge tiers.

3.2.1 T-Nodes

Every T-node has a unique name and it defines new concepts, roles, and individuals. For that purpose it contains an OLF lexicon and possibly an ACE text that states definitions on the new terms. A T-node may use the concepts, roles, and individuals from other T-nodes.

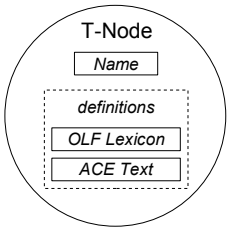


Figure 3.2: The Structure of a T-Node

The OLF lexicon contains only the entries for the introduced terms, but not the entries for the terms that are imported from other T-nodes. Thus for the parsing of the ACE text we need to merge the lexica from all the T-nodes that are needed. Every new term that is introduced has to be defined and we are not allowed to redefine terms from other T-nodes. Figure 3.2 shows the structure of a T-node.

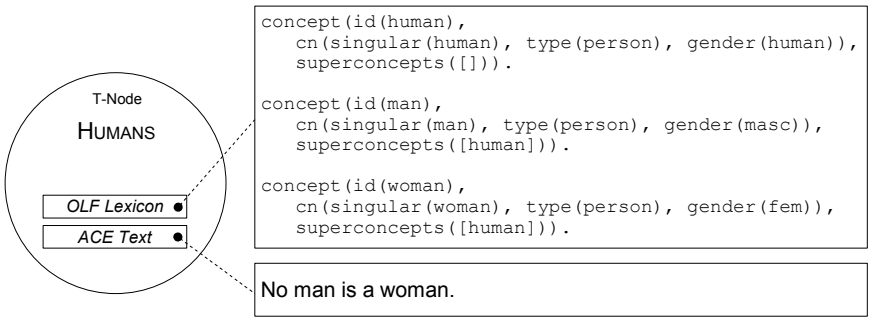


Figure 3.3: Example of a T-Node

If we want, for example, to create a T-node that defines the terms for representing humans then we first have to define a name for that node, e.g. HUMANS. Next, we have to create an OLF lexicon, which in our case might contain the concepts 'human', 'man', and 'woman'. The fact that 'man' and 'woman' are both subconcepts of 'human' can be expressed in the lexicon. If we want to define that the concepts 'man' and 'woman' are disjoint then we have to express it in ACE (e.g. 'no man is a woman'). Figure 3.3 illustrates this example.

3.2.2 K-Nodes

Like T-nodes, K-nodes have unique names. A K-node contains declarations about the domain. It can introduce new individuals, but it is not allowed to introduce new concepts or roles. Thus, a K-node consists of an ACE text and of an OLF lexicon that contains only individuals. This lexicon might be empty. Figure 3.4 shows the structure of a K-node.

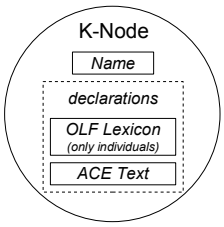


Figure 3.4: The Structure of a K-Node

Let us again have an example. We create a K-node that introduces two new individuals, 'John' and 'Mary', and thus we call this node JOHN&MARY. We have to create an OLF lexicon that defines the two individuals and then we can write ACE sentences that represent some knowledge about them. We use our T-node from above to declare that 'John is a man' and 'Mary is a woman'. Furthermore we state that there is a relationship between them that we express with 'John is the husband of Mary'[1]. Figure 3.5 shows this example of a K-node.

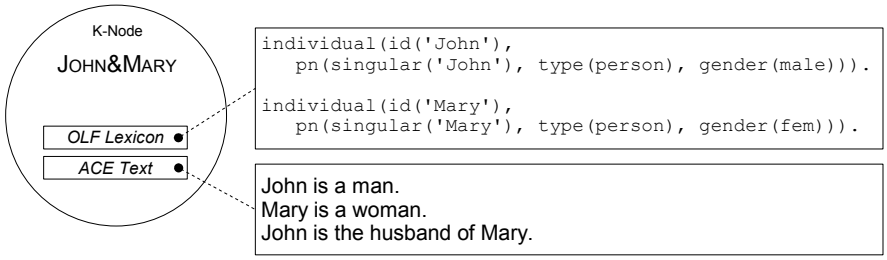


Figure 3.5: Example of a K-Node

[1]We assume that there exists another T-node that defines the role 'husband of'.

3.2.3 The Edges of the Graph

As already mentioned, the edges of the graph represent *uses*-relationships. That means that if a node uses one or more terms of another node then there is an edge pointing from the first to the latter. Every T-node belongs either to the base terminology tier or to the domain terminology tier; and every K-node belongs either to the common knowledge tier or to the additional knowledge tier. The edges have to satisfy the following properties.

- There are no edges from a T-node to a K-node.
- There are no edges from a T-node that belongs to the base terminology tier to a T-node that belongs to the domain terminology tier.
- There are no edges from a K-node that belongs to the common knowledge tier to a K-node that belongs to the additional knowledge tier.
- The edges describe a transitive relationship. If there is an edge form a node A to a node B and there is an edge from the node B to a node C then there is an edge from A to C. For the sake of clarity, we usually omit the edges that are implied by transitivity.
- The graph is a DAG, thus there are no cycles.

The graph structure is embedded in the four tier architecture. Figure 3.6 shows how the four tiers are related with the graph structure.

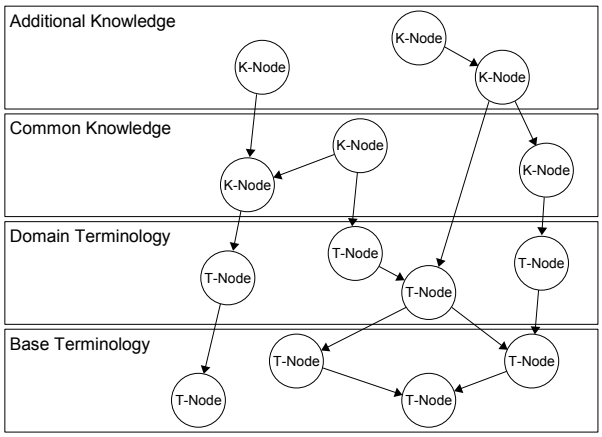


Figure 3.6: Four Tiers and Graph Structure of the ACE Ontology Architecture

Chapter 4

Building a Terminology

This chapter shows step by step how a terminology is built. First we create an exemplary base terminology that we will use in a second step to build a terminology for protein interactions upon.

4.1 An Exemplary Base Terminology

Before we can build a base terminology we have to think about the fundamental structure of our world. We have to be careful not to introduce any domain-specific aspects.

We divide our base terminology into three T-nodes that we call Base, Types, and ConPro. Figure 4.1 shows the structure and we will explain the three different nodes in the next sections. The detailed definition of the base terminology is shown in appendix A.1.

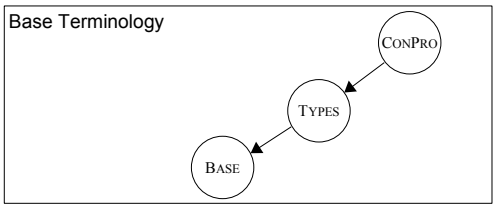


Figure 4.1: The Structure of the Base Terminology

4.1.1 The Node Base

The node Base introduces the five fundamental concepts 'entity', 'time', 'location', 'type', and 'instance' and the three roles 'part of', 'type of', and 'instance of'. Figure 4.2 shows the structure of Base.

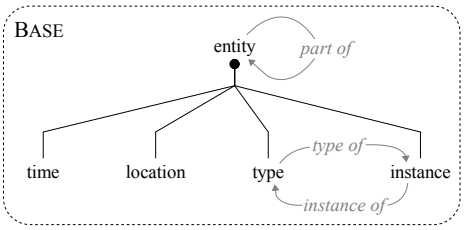


Figure 4.2: Structure of BASE

The concept 'entity' is the universal superconcept, i.e. every other concept is a subconcept of 'entity'[1]. 'Time' and 'location' stand for unmovable places in time and space that exist even if there are no other objects.

Let us have a closer look to the concepts 'type' and 'instance'. In any domain we have to distinguish between single objects of our world and types that represent an abstract view of multiple objects. 'Instance' stands for single objects (like items or processes) in our world and 'type', in contrast, stands for collections of instances that are similar in some respects. In order to have a consistent ontology, it is important to make this distinction.

The two roles 'type of' and 'instance of' make the link between types and instances. They assign types to instances and vice versa. A type can be assigned to several instances and an instance can be assigned to several types.

Finally we have the role 'part of'. This role allows to define compound objects that consist of other objects. 'Part of' is reflexive and transitive.

We are now done with our first T-node BASE which we can use to build other nodes upon.

4.1.2 The Node TYPES

With the node BASE we can express types, but we cannot structure the types in hierarchies. For that purpose we define the node TYPES that uses the node BASE. We introduce the roles 'subtype of' and 'supertype of' which are both mapped to ACE as *of*-constructs. These two roles allow us to define hierarchies of types. Figure 4.3 shows the structure.

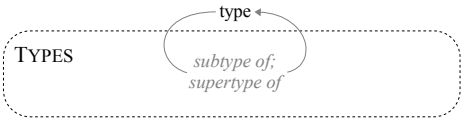


Figure 4.3: Structure of TYPES

The role 'subtype of' is the inverse role of 'supertype of'. If a type is a subtype of another type then that means: Every individual that belongs to the first type belongs as well to the second.

[1] In DL the universal superconcept is called 'top concept' and it is denoted by $\top$.

4.1.3 The Node ConPro

In a next step we want to provide more structure to types and instances. Thus we create the node ConPro which uses the node Types (and, as consequence, uses the node Base).

We want to distinguish between objects that endure through time (e.g. 'protein') and objects that occur (e.g. 'cell growth'). We call them continuants or processes, respectively. We do the same distinction on types and on instances. For that reason we introduce the four new concepts 'continuant', 'process', 'continuant-type', and 'process-type'. The concepts 'continuant' and 'process' are disjoint, and so are 'continuant-type' and 'process-type'. Figure 4.4 shows the structure.

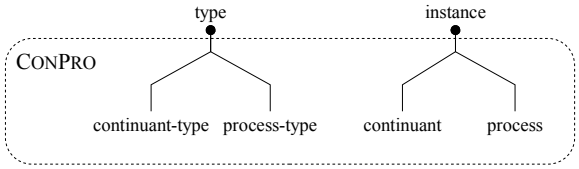


Figure 4.4: Structure of ConPro

Supertype- and subtype-relationships can only occur among continuant-types or among process-types, but not between them. Obviously every instance that belongs to a continuant-type has to be a continuant; and every instance that belongs to a process-type has to be a process.

All the three nodes of our base terminology are now defined and we are ready to use this base terminology for a specific domain.

Due to the graph structure of the terminology we can use only a part of the terminology, if we do not need all of it. If we want to use subtype-structures, for example, but we do not want to distinguish between continuants and processes, then we just use the nodes Base and Types. We do not need the node ConPro in this case, and thus it produces no overhead.

4.2 A Terminology for Protein Interactions

We are now ready to create a terminology for protein interactions. We will give only a coarse definition of the introduced terms. A thorough definition would exceed the scope of this thesis. For an excellent description of how to give a detailed definition of biomedical terms we recommend [17].

Our terminology for protein interaction uses the node ConPro that we defined in the previous section[2]. Again we divide our terminology into several T-nodes: Prot, Inter, Region, Go, ProtGo, and Prot2. Figure 4.5 shows the structure of the terminology for protein interactions. In the next sections we explain each of these nodes. The formal definition of the terminology is shown in appendix A.2.

[2]That means that we use all three nodes of the base terminology, since ConPro uses Types and Types uses Base.

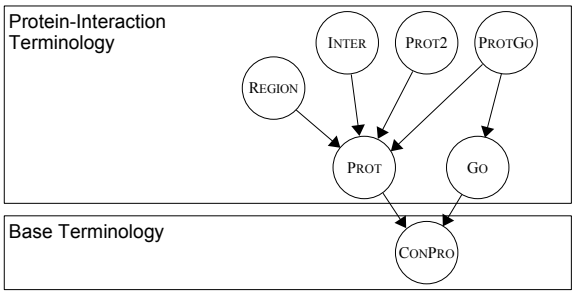


Figure 4.5: Structure of the Terminology for Protein Interactions

Furthermore, we will introduce some stub nodes that are not sufficiently structured and that we need just for demonstration purposes. For practical use we should give them a more detailed structure and we should split up each of them into multiple nodes.

4.2.1 The Node PROT

The node PROT defines the basic structure of proteins and it uses the node CONPRO of the base terminology.

Several proteins together can form a *protein-complex*, and proteins consist of different *regions* that are important for the interaction between them. Thus we introduce the concepts 'protein', 'protein-complex', 'region', 'protein-unit', and 'protein-component'. Each of them is expressed in ACE as a countable common noun.

We have to connect these concepts with the concepts that we defined in the base terminology. We have to decide whether the terms denote types or instances. In science we usually do not make statements about single objects of our world, but rather about types. Thus we define them as types. Figure 4.6 shows the structure of the node PROT.

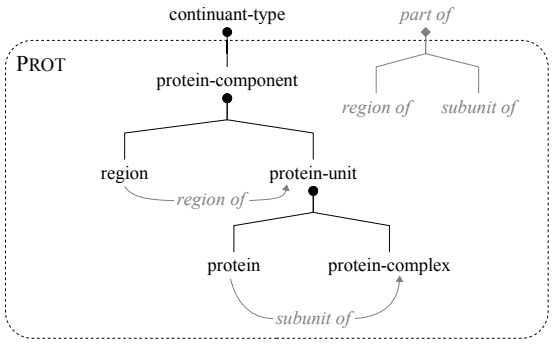


Figure 4.6: Structure of PROT

'Region' and 'protein-unit' are disjoint concepts and the same holds for 'protein' and 'protein-complex'. The roles 'region of' and 'subunit of' are subroles of the role 'part of' from the node BASE. With 'region of' we can express that a region is a part of a protein-unit and with 'subunit of' we can express that a protein is a part of a protein-complex. Both roles are expressed in ACE as *of*-constructs.

4.2.2 The Node GO

The node GO introduces the basic terms from the *Gene Ontology* (GO). The Gene Ontology is a controlled vocabulary for the description of genes and gene products[3]. We will use these terms to express information about proteins (e.g. where does a certain protein occur) and to express context information about protein interactions (e.g. to which process does the interaction of two proteins belong). The Gene Ontology consists of the three basic concepts 'cellular-component', 'molecular-function' and 'biological-process' which are mutually disjoint. Figure 4.7 shows the structure of this node.

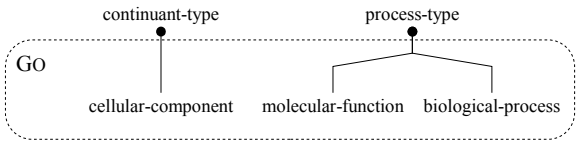


Figure 4.7: Structure of GO

This node contains only these three concepts, but not all the other terms from the Gene Ontology. For that reason we introduce the three stub nodes GOBP, GOCC, and GOMF that contain all the terms of the GO.

4.2.3 The Node INTER

The node INTER is the heart of our terminology and it allows to express interactions between proteins. Figure 4.8 shows the structure.

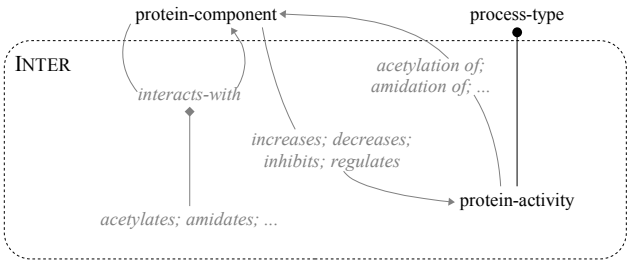


Figure 4.8: Structure of INTER

[3]See [9] and `http://www.geneontology.org/`.

Some protein interactions are expressed straightforward as transitive verbs, for example 'binds'; some of them are expressed as adverbs, for example 'interacts directly with'; and some of them are expressed as nouns that represent activities of a single protein (e.g. 'acetylation') linked with an auxiliary verb (i.e. 'increases', 'decreases', 'inhibits', and 'regulates'). Thus we can state, for example, the sentence 'a protein X decreases the acetylation of a protein Y'.

4.2.4 The Node ProtGo

The node ProtGo connects the node Prot with the node Go. It allows to express that a protein-component participates in a molecular-function or in a biological-process. Furthermore we can express that a protein-unit localizes to a cellular-component or that two protein-units co-localize. Figure 4.9 shows the structure of the node ProtGo.

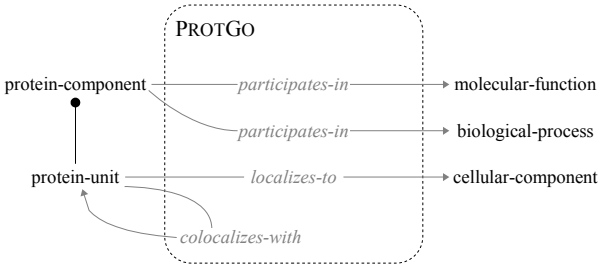


Figure 4.9: Structure of ProtGo

4.2.5 The Node Region

The node Region describes the inner structure of a protein, which is important for the interaction between proteins. The smallest component of a protein is a *residue*. A *multi-residue-region* is a structure that consists of several of such residues. 'Terminus', 'central-region', 'secondary-structure', and 'domain' are special cases of multi-residue-regions. A terminus is either a *n-terminus* or a *c-terminus*, and a secondary-structure is either a *alpha-helix* or a *beta-sheet*. Figure 4.10 shows the structure of this node.

4.2.6 The Node Prot2

The node Prot2 contains some additional terms for proteins. First we define 'receptor' as a subconcept of 'protein' and the role 'receptor of' that allows to express that a certain receptor belongs to another protein.

Furthermore we can characterize a protein-complex as a *dimer* or a *polymer* of a certain protein. For that reason we introduce the two roles 'dimer of' and 'polymer of'.

Finally we introduce two new roles that map among proteins: 'mutant of' and 'isoform of'. They allow us to declare mutants and isoforms of proteins. Figure 4.11 shows the structure.

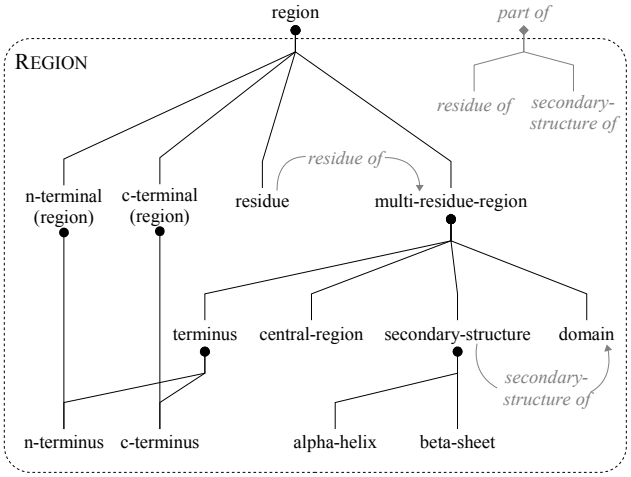


Figure 4.10: Structure of REGION

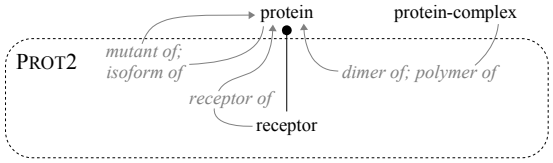


Figure 4.11: Structure of PROT2

4.2.7 The Stub Nodes

Finally we introduce eight stub nodes that we use for demonstration purposes: GOBP, GOCC, GOMF, PROTN, ANA, DIS, ORG, and SCIMETH.

The nodes GOBP, GOCC, and GOMF contain the terms that are extracted from the Gene Ontology and they build upon the node GO. They contain the terms about biological processes, cellular components, and molecular functions, respectively. We extracted the *is-a*-relationships from the GO to represent them as 'subtype of'-relations in ACE. We did not extract the *part-of*-relationships of GO, since they would belong to the knowledge tiers.

PROTN contains types of proteins that are used for expressing protein interactions and other facts about proteins.

The last four nodes are only used for context information for protein interactions. The node ANA contains terms that denote anatomical components which are extracted from MeSH (*Medical Subject Headings*)[4]. The nodes DIS and ORG are extracted from MeSH as well and contain terms about diseases and organisms respectively. The last node SCIMETH consists of terms that stand for scientific methods.

Since all these nodes are just stubs, they should get subdivided into multiple small nodes in order to get a clear structure.

We are now done with the terminology for protein interactions. Figure 4.12 shows the big picture of this terminology including the terms from the base terminology.

[4] `http://www.nlm.nih.gov/mesh/`

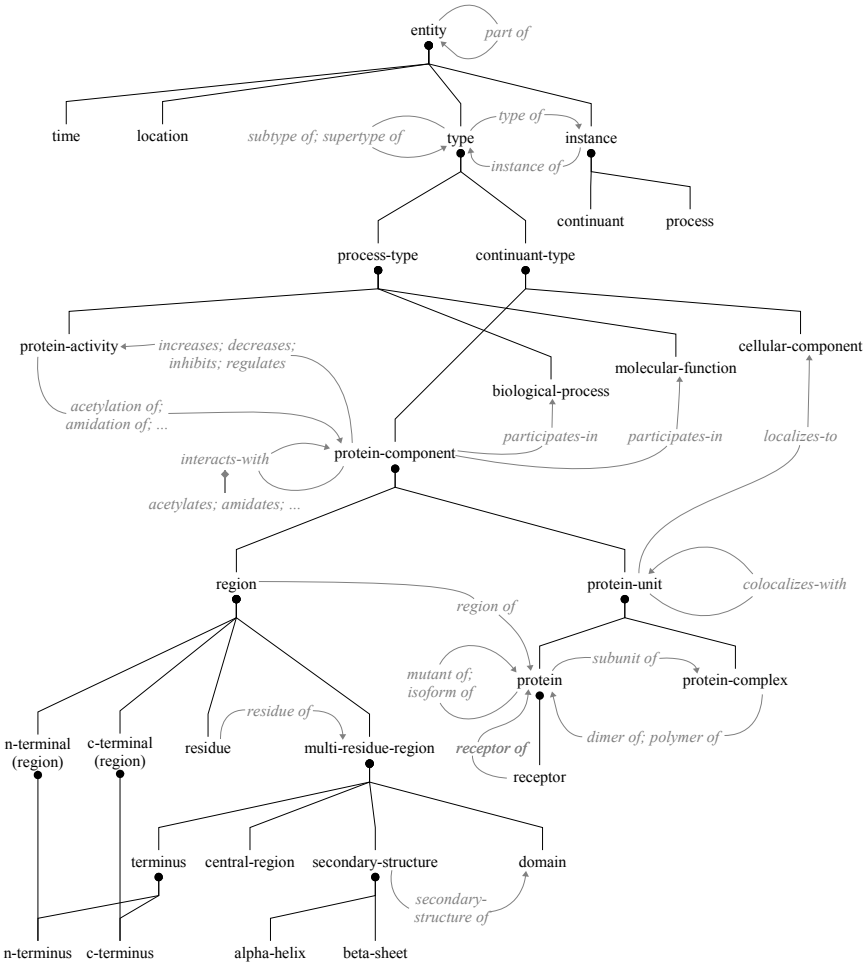


Figure 4.12: The Big Picture of the Terminology

Chapter 5

Expressing Knowledge

In the previous chapter we built a base terminology and a terminology for protein interactions. This chapter shows how we can express knowledge using these terminologies. For that purpose we use again ACE together with OLF lexica.

5.1 Accumulation of Knowledge

The modular tree structure of the ontology architecture allows us to extend the knowledge about a domain dynamically. Thus we do not have to specify all the knowledge at once. We can start with a small knowledge base and increase it step by step. This process can be seen as *knowledge assimilation* as described in [12].

Like terminologies, the knowledge is structured into nodes (which we call K-nodes). They are the smallest independent pieces of knowledge. If we want to express some new knowledge then we wrap it into one or more K-nodes and add them to one of the knowledge tiers of our ontology.

We do not allow to introduce new roles or concepts in the knowledge tiers. This can only be done on the level of the terminologies. But we are free to introduce new individuals which we address with propernames.

A K-node can use the concepts, roles, and individuals from other nodes. We do not have to specify these nodes explicitly, since we can just look for the places where the words are defined. Thus a K-node can use concepts, roles, and individuals from T-nodes and it can use individuals from other K-nodes.

5.1.1 State of Additional Knowledge

We have to think about the relationship between the common knowledge and the additional knowledge. For that reason we take a look at an arbitrary K-node that describes additional knowledge and its relationship to the common knowledge. There are four cases for the relationship between the information A of this K-node and the accumulated information B of the common knowledge [12]:

1. A is a logical consequence of B.

2. Part of B is logically implied by A together with the other part of B, i.e. $B = B_1 \cup B_2$ and B_2 is a logical consequence of $B_1 \cup \{A\}$.

3. A is inconsistent with B.

4. None of the relationships 1 – 3 hold.

On the basis of these four cases we can examine the relationship of additional knowledge to the common knowledge.

Additional knowledge that is the logical consequence of the common knowledge is useless in most cases, since there is no new information contained. In some situations it might be used as an extra confirmation of the truth of the corresponding part of the common knowledge.

If the additional knowledge fulfills the second condition, then this knowledge has the power to generalize the existing common knowledge. Such generalizations are very valuable since they allow to simplify the model of the world. But, unfortunately, this case is very hard to detect, since we have to check basically every possible split of the knowledge base B into two parts B_1 and B_2.

For additional knowledge that leads to inconsistency there are two possibilities: either the additional knowledge or the common knowledge is wrong. If we find out that the additional knowledge is wrong, then we can forget about it. Otherwise, if the common knowledge contains wrong information, we might have to rebuild the common knowledge by changing or removing some sentences. In some rare cases it might be better to keep the wrong common knowledge for the sake of a simple model. Since no model is perfect anyway, there will always be observations of the real world that are inconsistent with our model.

The forth case, where none of these relationships hold, is the most common one. The additional knowledge is not yet contained in the common knowledge, does not contradict to it, and has no generalization power. We expect most of the results of scientific papers to belong to this category.

5.1.2 The Paths of Knowledge

In this chapter, knowledge is described as a highly dynamic structure. On the one hand, there is new information that has to be integrated, and there is possibly knowledge that is falsified and has to be excluded from the ontology. On the other hand, we have to distinguish between validated knowledge and knowledge that is still uncertain. All that requires a dynamic management of knowledge. We introduce here the term *paths of knowledge*, that denotes the different kinds of changes of a single piece of knowledge. Figure 5.1 illustrates these paths of knowledge.

The path (1) stands for the inclusion of new knowledge that is not (yet) validated. Results from scientific papers could be such knowledge. At some point in time the knowledge might get validated or falsified. In the case of validation (3), we have to integrate it into the common knowledge, which might entail further transformations of other parts of the common knowledge (8). In the case of falsification (2), the knowledge has to be removed from the ontology. For unvalidated knowledge there can be corrections (7). Furthermore we can add new knowledge directly to the common knowledge (5), which is needed in particular for the initial construction of the common knowledge.

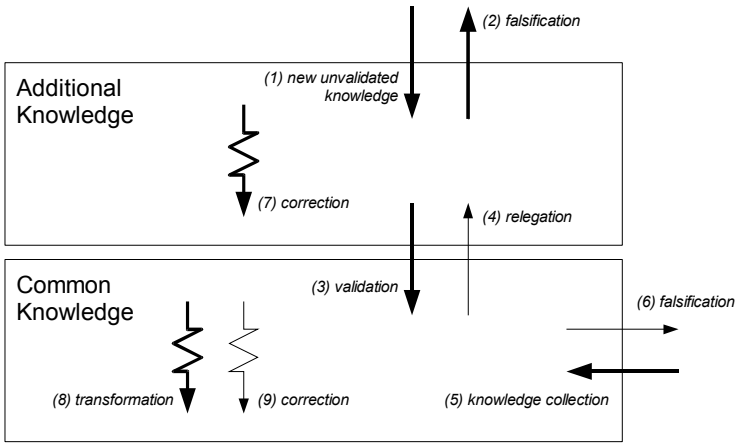


Figure 5.1: The Paths of Knowledge

In an ideal world, where we validate only true knowledge, these six paths would be sufficient. But in reality we have to take into account that we possibly validate knowledge that is wrong. Thus we need paths that allow us to correct such validations. We can correct the common knowledge (9) or we can remove parts of it (6). If there is a part of the common knowledge that we can not (yet) falsify, but there emerge doubts about its truth, then we can relegate it to the additional knowledge (4).

For the validation (3) of some additional knowledge A, the state of A – as described in the previous section – plays an important role. If A is the logical consequence of the common knowledge (the first case), then we do not have to change the common knowledge and A can be removed from the additional knowledge. The knowledge A was already known and thus the common knowledge remains unchanged.

If a part of the common knowledge is logically implied by some additional knowledge A together with the other part of the common knowledge (the second case), then A has the power to generalize the common knowledge. That means that we create redundancy if we integrate A into the common knowledge. Thus we can transform the common knowledge (e.g. to merge sentences) in order to eliminate this redundancy (8). Since this case is hard to detect (as we mentioned before), we probably cannot perform a complete check, and thus we have to live with a certain degree of redundancy in our knowledge base.

If we validate some knowledge that is inconsistent with the common knowledge (the third case) then we have to rebuild the common knowledge in order to preserve consistency. The rebuilding may include correction (9), transformation (8), falsification (6), and relegation (4) of parts of the common knowledge.

The fourth and most common case – none of the three relationships hold – is easy to handle. We can just add A to the common knowledge.

5.2 ACE Summaries

The initial goal of this thesis is to provide a formal language to summarize the results of scientific papers about protein interactions. We can finally take a closer look at this issue.

The language ACE allows us to write texts that look natural, but are in fact formal. Nonetheless ACE does not provide any help on translating natural language into some logical representation. ACE is designed for the creation of texts from scratch. If we have sentences in natural language that we want to express in ACE, then there is no other way than translating them *manually*.

5.2.1 ACE Summaries for 89 Selected Articles

Since we want to show how results of papers about protein interactions could have been written in ACE in the first place, we picked 89 articles from *Elsevier*-journals that concern protein interactions. Such articles mostly have a section called "Results" which is subdivided into subsections. The headings of these subsections are short descriptions of the corresponding results. It turned out that these headings are highly suitable for a manual translation into ACE. Note that we do this translation just for demonstration. It should *not* be the usual way to express the results first in natural language and then to translate them into ACE. Appendix B shows the headings and their representations in ACE.

The 89 articles contain 457 such headings. 184 of them are ignored, because they are not formulated as facts or because they contain information that is not about protein interactions.

total:		457	*(100%)*
ignored:	(not a fact)	87	*(19%)*
	(off-topic)	97	*(21%)*
used:		273	*(60%)*

After that we tried to translate the 273 remaining headings into ACE. For 154 of them there is a perfect match, which means that the complete information can be expressed in ACE. For another 62 headings only a part of the information is expressed, and for the remaining 57 headings there is no translation at all.

used:		273	*(100%)*
matched:	(perfect)	154	*(56%)*
	(partial)	62	*(23%)*
unmatched:		57	*(21%)*

Let us take a closer look at the reasons, why 119 headings cannot be rephrased in ACE perfectly. 56 of them could not be rephrased because their content is not covered by our model. These headings could be expressed with an extended model. Another 21 headings describe relations of relations, like the following heading from article #10 (see appendix B).

Kal-GEF1 activation of Pak does not require GEF activity.

In this case, there is a relation between two objects ('Pak activates Kal-GEF1') and this relation itself stands in another relation ('... does-not-require

GEF-activity'). We cannot express such structures in ACE – at least not in the usual way. In order to be able to express such relations of relations in a satisfying way, we would need to extend the language ACE.

Furthermore there are 11 headings with fuzzy statements and 31 headings that we could not understand.

not perfectly matched:	119	*(100%)*
not covered by our model:	56	*(47%)*
relations of relations:	21	*(18%)*
fuzzy:	11	*(9%)*
not understood:	31	*(26%)*

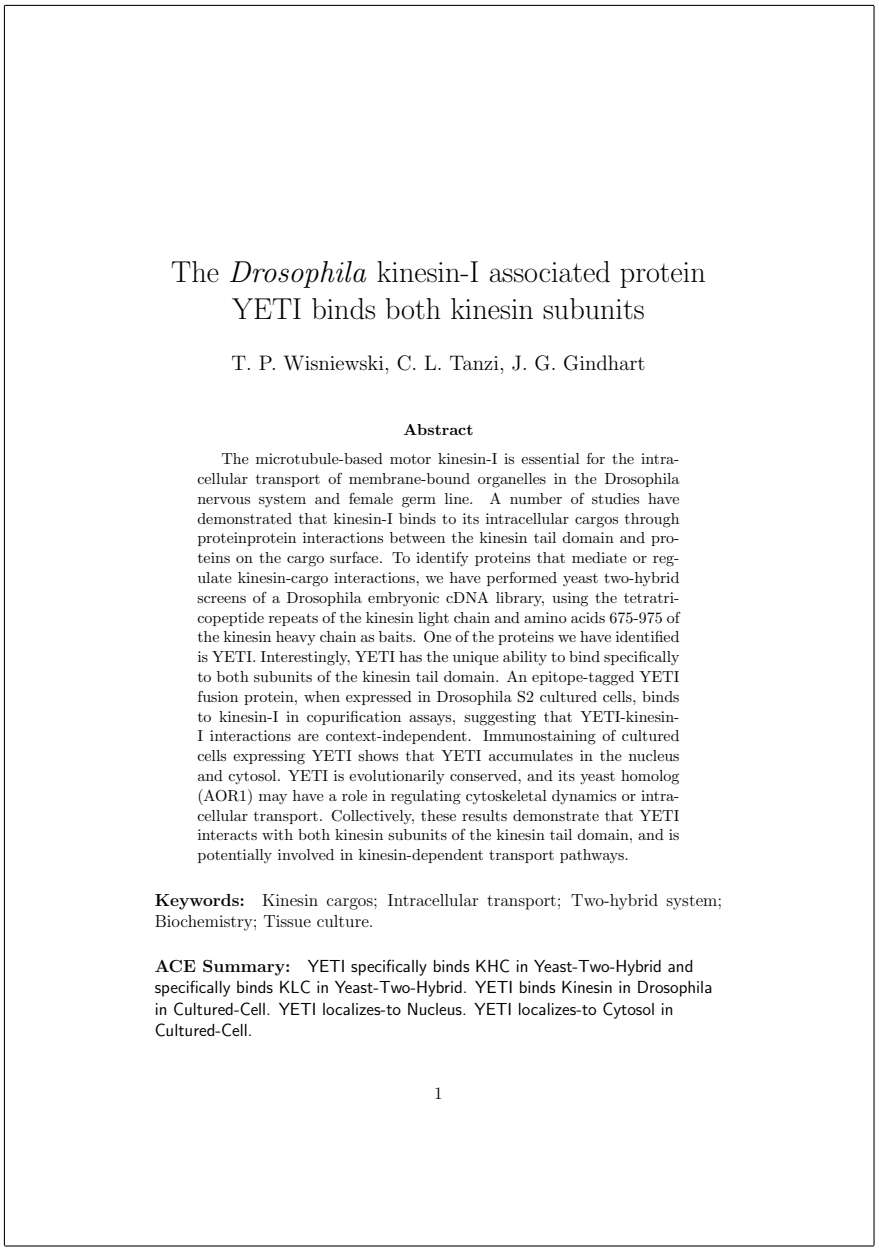
Thus, altogether we could rephrase 79% of the relevant headings, either partially or perfectly. This makes us confident that our approach is feasible for practical use. The reason, why 119 headings are not rephrased perfectly, is mostly our simple model and our lack of understanding. If we use a more detailed model and if we let the scientists express their own results in ACE, then we expect to be able to express much more than 79% of the results.

5.2.2 ACE Summary as an Integral Part of an Article

Since ACE looks like natural English, every reader of a scientific article is able to understand sentences in ACE. The ACE summary of the results can be an integral part of the article. Together with the abstract and a keyword list, it gives a concise insight into the content. Figure 5.2 shows how an article with an ACE summary could look like.

In contrast to the abstract, the ACE summary is readable by both, human and machines; and in contrast to the keyword list, the ACE summary does not only mention the objects of interest, but describes the relations among them.

Thus, every published article could be a contribution to a constantly growing ontology.



The *Drosophila* kinesin-I associated protein YETI binds both kinesin subunits

T. P. Wisniewski, C. L. Tanzi, J. G. Gindhart

Abstract

The microtubule-based motor kinesin-I is essential for the intracellular transport of membrane-bound organelles in the Drosophila nervous system and female germ line. A number of studies have demonstrated that kinesin-I binds to its intracellular cargos through proteinprotein interactions between the kinesin tail domain and proteins on the cargo surface. To identify proteins that mediate or regulate kinesin-cargo interactions, we have performed yeast two-hybrid screens of a Drosophila embryonic cDNA library, using the tetratricopeptide repeats of the kinesin light chain and amino acids 675-975 of the kinesin heavy chain as baits. One of the proteins we have identified is YETI. Interestingly, YETI has the unique ability to bind specifically to both subunits of the kinesin tail domain. An epitope-tagged YETI fusion protein, when expressed in Drosophila S2 cultured cells, binds to kinesin-I in copurification assays, suggesting that YETI-kinesin-I interactions are context-independent. Immunostaining of cultured cells expressing YETI shows that YETI accumulates in the nucleus and cytosol. YETI is evolutionarily conserved, and its yeast homolog (AOR1) may have a role in regulating cytoskeletal dynamics or intracellular transport. Collectively, these results demonstrate that YETI interacts with both kinesin subunits of the kinesin tail domain, and is potentially involved in kinesin-dependent transport pathways.

Keywords: Kinesin cargos; Intracellular transport; Two-hybrid system; Biochemistry; Tissue culture.

ACE Summary: YETI specifically binds KHC in Yeast-Two-Hybrid and specifically binds KLC in Yeast-Two-Hybrid. YETI binds Kinesin in Drosophila in Cultured-Cell. YETI localizes-to Nucleus. YETI localizes-to Cytosol in Cultured-Cell.

1

Figure 5.2: Article with ACE Summary: The frontpage of an article with an ACE summary could look like this. For this demonstration the article #22 of appendix B is used.

Chapter 6

Conclusions

6.1 The Benefits of our Approach

We showed in the preceding chapters what we need to do for expressing scientific results about protein interactions in ACE, and how it is done in detail. Now it is time to take a look at the benefits.

Imagine that all the scientific papers about protein interactions summarize their results in ACE. We could use these formal summaries to build up a dynamically growing knowledge base about protein interactions. On the basis of this knowledge base we would be able to answer many questions. We present now some examples of such questions.

Remembering that we decided to focus on a high degree of expressiveness, we have now to deal with a poor reasoning performance. That means that we probably need heuristics and simplifications and that we possibly cannot give complete answers to the questions that are described below. But we can argue that this problem is due to the inherent complexity of scientific results and is not made by our approach. We claim that even incomplete answers to these questions are a big advantage, compared to the possibilities we have today.

Are the results of a scientific paper consistent with the common knowledge and with other papers?

We can check, whether an ACE summary is consistent with the common knowledge. Usually this should be the case, since the common knowledge contains only knowledge that is supposed to be sure. If a paper contains results that are inconsistent with the common knowledge, then this can be seen as an appeal against the common knowledge.

Without a formal declaration of common knowledge and scientific results, it is impossible to check for consistency. Probably there exist many scientific papers that contain results which are inconsistent with some common knowledge. But since it can be very difficult to find out, neither the author nor the readers might realize the special status of the results.

In the same way we can check, whether there exist papers that contradict a certain paper. That would mean that different researchers claim contradictory results. Being aware of such a contradiction might lead to a dialogue between the corresponding scientists, which might entail better and consistent results.

Are the results of a scientific paper already known?

With our formal approach we can check whether a certain result is already known. Results that are already considered common knowledge are not worth to be described as results of scientific papers. Thus it is very valuable to be able to run a check, whether a certain result is already contained in the common knowledge or not.

Furthermore a researcher might want to check, whether there exists scientific literature that has arrived at the same results. Altogether our approach would help the researchers to save a lot of time, since they would not need to search "manually" for the relevant literature.

Is there a known answer for a certain question?

If someone – researcher or not – has a specific question about the domain (e.g. protein interactions), then we would be able to extract automatically an answer[1]. Such an answer-extraction can consider only the common knowledge, or it can include every known scientific paper.

What is known about a certain object of interest?

In some cases we do not want to ask a specific question, but we rather want to get an overview of a single object of interest (e.g. the protein IRAK2). If we ask for information about such an object then we might get something like[2]

type	IRAK2
supertypes	IRAK – Protein – Molecule
subtypes	IRAK2a, IRAK2b
interacts directly with	BTG2, XDH, Mcm3, MAX
interacts indirectly with	BCL2, Indo, Cckbr, HPCA, ID1, Ep300
phosphorylates	XDH
colocalizes with	BTG2, HPCA
localizes to	Membrane
participates in	Cell-Growth, Signal-Transduction

Again we have the option to consider either only the common knowledge or to include all known scientific papers. The idea of such an overview could be used for a dynamic hypertext representation. This would allow us to navigate through the whole knowledge base, e.g. with an ordinary web browser. New papers that are submitted can be integrated *automatically* and thus such a web interface would be always up-to-date.

How are some objects of interest related?

Instead of focusing on one single object, we might want to have an overview of the interrelations of a certain group of objects. We could extract, for example, the *interacts-with*-relations of all proteins and use this data for further examination, like the detection of clusters or hot-spots. Such examinations are already common in the research on proteins, but only with restricted data. With our approach we could consider every interaction that has been published.

[1] ACE allows to formulate questions which could be used for queries on the knowledge base.
[2] this example is purely fictitious

6.2 Summary

The goal of this thesis was to show, how the results of scientific papers on protein interactions can be expressed in ACE, in order to make them machine-readable. For that purpose we decided to create an ontology for protein interactions, which should serve as basis for the descriptions of the results. We adopted the basic elements of Descriptions Logics – individuals, concepts, and roles – for our ontology in ACE, but we retained the expressiveness of first-order logic.

For the specification of an ontology in ACE we needed to introduce a new lexicon format: Ontology Lexicon Format. This format is used to define the basic structure of the ontology and the representations in ACE. For the translation of ACE into the logical representation (i.e. the DRS), we used the existing parser APE together with some additional tools.

In order to give a clear structure to the ontology, we defined an ontology architecture for ACE. Basically such an ontology is divided into four tiers: two terminology tiers and two knowledge tiers. The terminology tiers define the terms and their interrelation, and we distinguish between a domain-independent and a domain-specific terminology. One of the two knowledge tiers defines the common knowledge, whereas the other defines additional knowledge that is uncertain.

For a dynamic management of the ontology we defined a graph structure for the ontology. This allowed us to handle knowledge as a highly dynamic structure and we introduced the term *paths of knowledge*, that stands for the kinds of changes that a piece of knowledge undergoes.

We defined an ontology for protein interactions and demonstrated how it can be used for the expression of scientific results. For that purpose we took 89 articles and demonstrated how the results could have been written in ACE. We could express 56% of the results perfectly in ACE, and another 23% can be expressed partially. These results make us quite confident that our approach is feasible for practical use.

6.3 Future Work

This thesis suggests an approach of using controlled natural language for making the results of scientific papers readable and – to some degree – understandable by computers. But in order to achieve this goal, there is still a lot of work to do. We point out here some of the major tasks that have to be performed.

Writing Assistance Tool. First of all, we need a writing assistance tool as it is sketched in section 2.3, in order to support the authors of scientific papers in the creation of ACE summaries.

Terminology Definition System. Next, we need a tool that allows to create and maintain the terminological definitions of an ontology. This tool is used by system experts, i.e. the system administrators.

Knowledge Management System. For the collection and management of knowledge on the basis of an existing ontology, we need a knowledge manage-

ment tool. Among other tasks, this tool is responsible for the consistency of the knowledge base.

Creation of an Ontology. We have to create an ontology of the corresponding domain in order to be able to collect knowledge about this domain. In this thesis we showed how it could be done.

Collection of Common Knowledge. The collection of common knowledge is an important and demanding task. We have to map the knowledge on the structures that are defined by the ontology.

Commitment among Scientists. Besides all these technical requirements, there are also political requisites. There must be a commitment among the scientists of the corresponding field of research – or at least among a big part of them – that every scientific article has to summarize its results in ACE. If such a summary is optional then there is little hope that it gets established.

Altogether we can say that there is a lot of work to do, and we do not expect to have a working system in the near future. But due to the immense benefits such a system would bring along, we belief in the great potential of our approach.

Appendix A

Terminology Definitions

A.1 Definition of the Base Terminology

The Node BASE

Lexicon

```
concept(id(entity),
        cn(singular(entity), type(unspecified), gender(neutr)),
        superconcepts([])).

concept(id(time),
        cn(singular(time), type(time), gender(neutr)),
        superconcepts([entity])).

concept(id(location),
        cn(singular(location), type(unspecified), gender(neutr)),
        superconcepts([entity])).

concept(id(type),
        cn(singular(type), type(unspecified), gender(neutr)),
        superconcepts([entity])).

role(id('type-of'),
     cn(ref(type)),
     superroles([]),
     domain(type),
     range(instance)).

concept(id(instance),
        cn(singular(instance), type(unspecified), gender(neutr)),
        superconcepts([entity])).

role(id('instance-of'),
     cn(ref(instance)),
     superroles([]),
     domain(instance),
     range(type)).

role(id('part-of'),
     cn(singular(part), type(unspecified), gender(neutr)),
     superroles([]),
     domain(entity),
     range(entity)).
```

Definitions

Everything is an entity.

No time is a location. No time is a type. No time is an instance. No location is a type. No location is an instance. No type is an instance.

If an instance Y is an instance of a type X then X is a type of Y. If a type X is a type of an instance Y then Y is an instance of X.

Everything is a part of itself.

If an entity X is a part of an entity Y that is a part of an entity Z then X is a part of Z.

If an entity X is a part of a time then X is a time. If an entity X is a part of a location then X is a location. If an entity X is a part of a type then X is a type. If an entity X is a part of an instance then X is an instance.

The Node TYPES

Lexicon

```
role(id('subtype-of'),
     cn(singular(subtype), type(unspecified), gender(neutr)),
     superroles([]),
     domain(type),
     range(type)).

role(id('supertype-of'),
     cn(singular(supertype), type(unspecified), gender(neutr)),
     superroles([]),
     domain(type),
     range(type)).
```

Definitions

If a type X is a subtype of a type Y then Y is a supertype of X. If a type X is a supertype of a type Y then Y is a subtype of X.

If an instance X is an instance of a type that is a subtype of a type Y then X is an instance of Y.

The Node CONPRO

Lexicon

```
concept(id(continuant),
        cn(singular(continuant), type(unspecified), gender(neutr)),
        superconcepts([instance])).

concept(id(process),
        cn(singular(process), type(unspecified), gender(neutr)),
        superconcepts([instance])).

concept(id('continuant-type'),
        cn(singular('continuant-type'), type(unspecified), gender(neutr)),
        superconcepts([type])).

concept(id('process-type'),
        cn(singular('process-type'), type(unspecified), gender(neutr)),
        superconcepts([type])).
```

Definitions

No continuant is a process. No continuant-type is a process-type.

If a type X is a subtype of a continuant-type then X is a continuant-type. If a type X is a subtype of a process-type then X is a process-type.

If an instance X is an instance of a continuant-type then X is a continuant. If an instance X is an instance of a process-type then X is a process.

A.2 Definition of the Terminology for Protein Interactions

The Node PROT

Lexicon

```
concept(id('protein-component'),
        cn(singular('protein-component'), type(object), gender(neutr)),
        superconcepts(['continuant-type'])).

concept(id(region),
        cn(singular(region), type(object), gender(neutr)),
        superconcepts(['protein-component'])).

role(id('region-of'),
     cn(ref(region)),
     superroles(['part-of']),
     domain(region),
     range(protein)).

concept(id('protein-unit'),
        cn(singular('protein-unit'), type(object), gender(neutr)),
        superconcepts(['protein-component'])).

concept(id(protein),
        cn(singular(protein), type(object), gender(neutr)),
        superconcepts(['protein-component'])).

concept(id('protein-complex'),
        cn(singular('protein-complex'), type(object), gender(neutr)),
        superconcepts(['protein-unit'])).

role(id('subunit-of'),
     cn(singular(subunit), type(object), gender(neutr)),
     superroles(['part-of']),
     domain(protein),
     range('protein-complex')).
```

Definitions

No region is a protein-unit. No protein is a protein-complex.

The Node GO

Lexicon

```
concept(id('cellular-component'),
        cn(singular('cellular-component'), type(object), gender(neutr)),
        superconcepts(['continuant-type'])).

concept(id('molecular-function'),
        cn(singular('molecular-function'), type(unspecified), gender(neutr)),
        superconcepts(['process-type'])).

concept(id('biological-process'),
        cn(singular('biological-process'), type(unspecified), gender(neutr)),
        superconcepts(['process-type'])).
```

Definitions

No molecular-function is a biological-process.

If an entity X is a part of a cellular-component then X is a cellular-component. If an entity X is a part of a molecular-function then X is a molecular-function. If an entity X is a part of a biological-process then X is a biological-process.

The Node INTER

Lexicon

```
role(id('interacts-with'),
     tv(third_singular(interacts), third_plural(interact),
        phrasal_particle(''), direct_preposition(with)),
     superroles([]),
     domain('protein-component'),
     range('protein-component'),
     context([prep(in, 'cellular-component'), prep(for, 'molecular-function'),
              prep(in, 'biological-process'), prep(in, 'scientific-method'),
              prep(in, 'anatomical-component'), prep(in, 'disease'),
              prep(in, 'organism')])).

role(id(acetylates),
     tv(third_singular(acetylates), third_plural(acetylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(amidates),
     tv(third_singular(amidates), third_plural(amidate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(associates),
     tv(third_singular(associates), third_plural(associate),
        phrasal_particle(''), direct_preposition(with)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(autophosphorylates),
     tv(third_singular(autophosphorylates), third_plural(autophosphorylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(binds),
     tv(third_singular(binds), third_plural(bind),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(cleaves),
     tv(third_singular(cleaves), third_plural(cleave),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(deacetylates),
     tv(third_singular(deacetylates), third_plural(deacetylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(deamidates),
     tv(third_singular(deamidates), third_plural(deamidate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(deglycosylates),
     tv(third_singular(deglycosylates), third_plural(deglycosylate),
        phrasal_particle(''), direct_preposition('')),
```

```
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(degrades),
     tv(third_singular(degrades), third_plural(degrade),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(demethylates),
     tv(third_singular(demethylates), third_plural(demethylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(deneddylates),
     tv(third_singular(deneddylates), third_plural(deneddylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(dephosphorylates),
     tv(third_singular(dephosphorylates), third_plural(dephosphorylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(desumoylates),
     tv(third_singular(desumoylates), third_plural(desumoylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(deubiquitinates),
     tv(third_singular(deubiquitinates), third_plural(deubiquitinate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(farnesylates),
     tv(third_singular(farnesylates), third_plural(farnesylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(gernaylgernanylates),
     tv(third_singular(gernaylgernanylates), third_plural(gernaylgernanylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(glycosylates),
     tv(third_singular(glycosylates), third_plural(glycosylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(homodimerizes),
     tv(third_singular(homodimerizes), third_plural(homodimerize),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).
```

```
role(id(hydroxylates),
     tv(third_singular(hydroxylates), third_plural(hydroxylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(internalizes),
     tv(third_singular(internalizes), third_plural(internalize),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(methylates),
     tv(third_singular(methylates), third_plural(methylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(mobilizes),
     tv(third_singular(mobilizes), third_plural(mobilize),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(neddylates),
     tv(third_singular(neddylates), third_plural(neddylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(nitrates),
     tv(third_singular(nitrates), third_plural(nitrate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(oxidizes),
     tv(third_singular(oxidizes), third_plural(oxidize),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(phosphorylates),
     tv(third_singular(phosphorylates), third_plural(phosphorylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(prenylates),
     tv(third_singular(prenylates), third_plural(prenylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(reduces),
     tv(third_singular(reduces), third_plural(reduce),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(sumoylates),
     tv(third_singular(sumoylates), third_plural(sumoylate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
```

```
     domain('protein-component'),
     range('protein-component')).

role(id(ubiquitinates),
     tv(third_singular(ubiquitinates), third_plural(ubiquitinate),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id(polymerizes),
     tv(third_singular(polymerizes), third_plural(polymerize),
        phrasal_particle(''), direct_preposition('')),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('specifically-interacts-with'),
     adv(adverb(specifically), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('selectively-interacts-with'),
     adv(adverb(selectively), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('directly-interacts-with'),
     adv(adverb(directly), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('indirectly-interacts-with'),
     adv(adverb(indirectly), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('hydrophobically-interacts-with'),
     adv(adverb(hydrophobically), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('physically-interacts-with'),
     adv(adverb(physically), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('functionally-interacts-with'),
     adv(adverb(functionally), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('stably-interacts-with'),
     adv(adverb(stably), type(manner)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('interacts-with-in-vivo'),
     adv(adverb('in-vivo'), type(location)),
     superroles(['interacts-with']),
     domain('protein-component'),
     range('protein-component')).

role(id('interacts-with-in-vitro'),
     adv(adverb('in-vitro'), type(location)),
     superroles(['interacts-with']),
```

```
     domain('protein-component'),
     range('protein-component')).

concept(id('protein-activity'),
        cn(singular('protein-activity'), type(unspecified), gender(neutr)),
        superconcepts(['process-type'])).

role(id('acetylation-of'),
     cn(singular(acetylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('amidation-of'),
     cn(singular(amidation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('autophosphorylation-of'),
     cn(singular(autophosphorylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('cleavage-of'),
     cn(singular(cleavage), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('deacetylation-of'),
     cn(singular(deacetylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('degradation-of'),
     cn(singular(degradation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('dephosphorylation-of'),
     cn(singular(dephosphorylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('depolymerization-of'),
     cn(singular(depolymerization), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('glycosylation-of'),
     cn(singular(glycosylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('hydroxylation-of'),
     cn(singular(hydroxylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('internalization-of'),
     cn(singular(internalization), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('methylation-of'),
```

```
     cn(singular(methylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('neddylation-of'),
     cn(singular(neddylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('nitration-of'),
     cn(singular(nitration), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('nitrosylation-of'),
     cn(singular(nitrosylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('oxidation-of'),
     cn(singular(oxidation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('phosphorylation-of'),
     cn(singular(phosphorylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('polymerization-of'),
     cn(singular(polymerization), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('prenylation-of'),
     cn(singular(prenylation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('pro-cleavage-of'),
     cn(singular('pro-cleavage'), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('reduction-of'),
     cn(singular(reduction), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('release-of'),
     cn(singular(release), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('secretion-of'),
     cn(singular(secretion), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('sumoylation-of'),
     cn(singular(sumoylation), type(unspecified), gender(neutr)),
     superroles([]),
```

```
     domain('protein-activity'),
     range('protein-component')).

role(id('ubiquitination-of'),
     cn(singular(ubiquitination), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('activity-of'),
     cn(singular(activity), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('efflux-of'),
     cn(singular(efflux), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('expression-of'),
     cn(singular(expression), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('influx-of'),
     cn(singular(influx), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('mobilization-of'),
     cn(singular(mobilization), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('stabilization-of'),
     cn(singular(stabilization), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('trafficking-of'),
     cn(singular(trafficking), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id('translocation-of'),
     cn(singular(translocation), type(unspecified), gender(neutr)),
     superroles([]),
     domain('protein-activity'),
     range('protein-component')).

role(id(increases),
     tv(third_singular(increases), third_plural(increase),
        phrasal_particle(''), direct_preposition('')),
     superroles([]),
     domain('protein-component'),
     range('protein-activity'),
     context([prep(in, 'cellular-component'), prep(for, 'molecular-function'),
              prep(in, 'biological-process'), prep(in, 'scientific-method'),
              prep(in, 'anatomical-component'), prep(in, 'disease'),
              prep(in, 'organism')])).

role(id(decreases),
     tv(third_singular(decreases), third_plural(decrease),
        phrasal_particle(''), direct_preposition('')),
     superroles([]),
     domain('protein-component'),
     range('protein-activity'),
```

```
     context([prep(in, 'cellular-component'), prep(for, 'molecular-function'),
              prep(in, 'biological-process'), prep(in, 'scientific-method'),
              prep(in, 'anatomical-component'), prep(in, 'disease'),
              prep(in, 'organism')])).

role(id(inhibits),
     tv(third_singular(inhibits), third_plural(inhibit),
        phrasal_particle(''), direct_preposition('')),
     superroles([]),
     domain('protein-component'),
     range('protein-activity'),
     context([prep(in, 'cellular-component'), prep(for, 'molecular-function'),
              prep(in, 'biological-process'), prep(in, 'scientific-method'),
              prep(in, 'anatomical-component'), prep(in, 'disease'),
              prep(in, 'organism')])).

role(id(regulates),
     tv(third_singular(regulates), third_plural(regulate),
        phrasal_particle(''), direct_preposition('')),
     superroles([]),
     domain('protein-component'),
     range('protein-activity'),
     context([prep(in, 'cellular-component'), prep(for, 'molecular-function'),
              prep(in, 'biological-process'), prep(in, 'scientific-method'),
              prep(in, 'anatomical-component'), prep(in, 'disease'),
              prep(in, 'organism')])).
```

The Node ProtGo

Lexicon

```
role(id('participates-in'),
     tv(third_singular(participates), third_plural(participate),
        phrasal_particle(''), direct_preposition('in')),
     superroles([]),
     domain('protein-component'),
     range('process-type')).

role(id('localizes-to'),
     tv(third_singular(localizes), third_plural(localize),
        phrasal_particle(''), direct_preposition('to')),
     superroles([]),
     domain('protein-unit'),
     range('cellular-component')).

role(id('colocalizes-with'),
     tv(third_singular(colocalizes), third_plural(colocalize),
        phrasal_particle(''), direct_preposition('with')),
     superroles([]),
     domain('protein-unit'),
     range('protein-unit')).
```

Definitions

If something participates-in an entity X then X is a molecular-function or X is a biological-process.

The Node Region

Lexicon

```
concept(id('n-terminal-region'),
        adj(positive('n-terminal')),
        superconcepts([region])).

concept(id('c-terminal-region'),
        adj(positive('c-terminal')),
        superconcepts([region])).

concept(id(residue),
```

```
        cn(singular(residue), type(object), gender(neutr)),
        superconcepts([region])).

role(id('residue-of'),
     cn(ref(residue)),
     superroles(['part-of']),
     domain(residue),
     range('multi-residue-region')).

concept(id('multi-residue-region'),
        cn(singular('multi-residue-region'), type(object), gender(neutr)),
        superconcepts([region])).

concept(id(terminus),
        cn(singular(terminus), type(object), gender(neutr)),
        superconcepts(['multi-residue-region'])).

concept(id('central-region'),
        cn(singular('central-region'), type(object), gender(neutr)),
        superconcepts(['multi-residue-region'])).

concept(id('secondary-structure'),
        cn(singular('secondary-structure'), type(object), gender(neutr)),
        superconcepts(['multi-residue-region'])).

role(id('secondary-structure-of'),
     cn(ref('secondary-structure')),
     superroles(['part-of']),
     domain('secondary-structure'),
     range('domain')).

concept(id(domain),
        cn(singular(domain), type(object), gender(neutr)),
        superconcepts(['multi-residue-region'])).

concept(id('n-terminus'),
        cn(singular('n-terminus'), type(object), gender(neutr)),
        superconcepts(['n-terminal-region', terminus])).

concept(id('c-terminus'),
        cn(singular('c-terminus'), type(object), gender(neutr)),
        superconcepts(['c-terminal-region', terminus])).

concept(id('alpha-helix'),
        cn(singular('alpha-helix'), type(object), gender(neutr)),
        superconcepts(['secondary-structure'])).

concept(id('beta-sheet'),
        cn(singular('beta-sheet'), type(object), gender(neutr)),
        superconcepts(['secondary-structure'])).
```

Definitions

No n-terminal region is a c-terminal region. No residue is a multi-residue-region.

No terminus is a central-region. No secondary-structure is a domain.

No alpha-helix is a beta-sheet.

The Node PROT2

Lexicon

```
concept(id(receptor),
        cn(singular(receptor), type(object), gender(neutr)),
        superconcepts([protein])).

role(id('receptor-of'),
     cn(ref(receptor)),
     superroles([]),
     domain(receptor),
     range(protein)).
```

```
role(id('dimer-of'),
     cn(singular(dimer), type(object), gender(neutr)),
     superroles([]),
     domain('protein-complex'),
     range(protein)).

role(id('polymer-of'),
     cn(singular(polymer), type(object), gender(neutr)),
     superroles([]),
     domain('protein-complex'),
     range(protein)).

role(id('isoform-of'),
     cn(singular(isoform), type(object), gender(neutr)),
     superroles([]),
     domain(protein),
     range(protein)).

role(id('mutant-of'),
     cn(singular(mutant), type(object), gender(neutr)),
     superroles([]),
     domain(protein),
     range(protein)).
```

Appendix B

Article Headings

This appendix shows how the results of 89 selected articles from *Elsevier*-journals could have been written in ACE. For that reason the headings of the section "Results" of these articles are translated into ACE, if possible. Some headings are ignored; they are indicated as follows.

not a fact	The heading is ignored, because it is not a fact.
off-topic	The heading is ignored, because it is not about protein interactions.

For every other heading the degree of matching is indicated.

perfect	The heading is matched perfectly (the complete information is translated).
perfect (r)	The heading is matched perfectly, but the information is already contained in a previous sentence.
partial	The heading is partially matched (some information is translated, and some is not).
partial (r)	The heading is partially matched, but the information is already contained in a previous sentence.
no match	The heading is not matched (the information is not translated at all).

For the headings, that are not perfectly matched, (i.e. the headings that are indicated with *partial*, *partial (r)*, or *no match*) the reason for the non-perfect matching is declared as follows.

[1]	not covered by our model
[2]	relation of relation
[3]	fuzzy
[4]	not understood

Article #1

AUTHORS: *Kanamori M, Kai C, Hayashizaki Y, Suzuki H.*
TITLE: *NF-kappaB activator Act1 associates with IL-1/Toll pathway adaptor molecule TRAF6.*
PMID: *12459498.*

Heading	ACE
Interaction of Act1 with TRAF6.	*perfect:* Act1 interacts-with TRAF6.
Act1 specifically interacts with TRAF6 through the TRAF domain.	*perfect:* Act1 specifically interacts-with a TRAF-domain of TRAF6.
Reporter gene analysis using the expression vector for Act1.	*not a fact.*

Article #2

AUTHORS: *Strelow A, Kollewe C, Wesche H.*
TITLE: *Characterization of Pellino2, a substrate of IRAK1 and IRAK4.*
PMID: *12860405.*

Heading	ACE
Interaction of Pellino2 with IRAKs.	*perfect:* Pellino2 interacts-with IRAK.
Pellino2 phosphorylation by IRAKs.	*perfect:* IRAK phosphorylates Pellino2.
Functional characterization of Pellino2.	*not a fact.*

Article #3

AUTHORS: *Bongiorno-Borbone L, Kadare G, Benfenati F, Girault JA.*
TITLE: *FAK and PYK2 interact with SAP90/PSD-95-Associated Protein-3.*
PMID: *16202977.*

Heading	ACE
SAPAP3 interacts with FAK in yeast two-hybrid.	*perfect:* SAPAP3 interacts-with FAK in Yeast-Two-Hybrid.
Interaction between FAK and SAPAP3 in GST pull-down assays.	*perfect:* SAPAP3 interacts-with FAK in GST-Pull-Down-Assay.
Interaction between SAPAP3 and PYK2.	*perfect:* SAPAP3 interacts-with PYK2.
Subcellular expression of SAPAP3 and co-distribution with FAK and PYK2.	*off topic.*

Article #4

AUTHORS: *Lahiri S, Pulakat L, Gavini N.*
TITLE: *Functional NifD-K fusion protein in Azotobacter vinelandii is a homodimeric complex equivalent to the native heterotetrameric MoFe protein.*
PMID: *16202390.*

Heading	ACE
Detection of interaction between NifD-K fusion protein units.	*perfect:* A subunit of NifD-K interacts-with a subunit of NifD-K.
Interaction between the NifD-K fusion proteins is comparable to the interaction between the $\beta - \beta$ subunits of the native MoFe protein.	*no match*[3].

Article #5

AUTHORS: *Veyron-Churlet R, Bigot S, Guerrini O, Verdoux S, Malaga W, Daffe M, Zerbib D.*
TITLE: *The biosynthesis of mycolic acids in Mycobacterium tuberculosis relies on multiple specialized elongation complexes interconnected by specific protein-protein interactions.*
PMID: *16213523.*

Heading	ACE
Yeast two-hybrid (Y2H) analysis of protein-protein interactions between mtFabD and the FAS-II components.	*not a fact.*
The mtFabD protein is part of the core of the FAS-II complex	*partial*[1]*:* MtFabD is a subunit of FAS-II.
The methyltransferases MmaAs interact with the FAS-II proteins in Y2H analysis.	*perfect:* MmaAs interacts-with FAS-II in Yeast-Two-Hybrid.
Co-IP analysis of MmaAs interactions with the FAS-II proteins.	*not a fact.*
The MmaA4 interaction is specific with KasA and KasB and not with mtFabH.	*perfect:* MmaA4 specifically interacts-with KasA and specifically interacts-with KasB and does not specifically interact-with MtFabH.
The terminal condensing enzyme Pks13 interacts with the FAS-II complex proteins.	*perfect:* Pks13 interacts-with FAS-II.
The architecture of an elongationmodification complex for the biosynthesis of mycolic acids.	*not a fact.*

Article #6

Authors: *Masumi A, Aizaki H, Suzuki T, DuHadaway JB, Prendergast GC, Komuro K, Fukazawa H.*
Title: *Reduction of hepatitis C virus NS5A phosphorylation through its interaction with amphiphysin II.*
PMID: *16139795.*

Heading	ACE
NS5A associates with amphiphysin II in HeLa cells.	*perfect:* NS5A associates-with Amphiphysin-II in Hela-Cell.
The interaction with amphiphysin II SH3 inhibits NS5A phosphorylation.	*perfect:* A SH3-domain of Amphiphysin-II inhibits the phosphorylation of NS5A.

Article #7

Authors: *Rajendran KS, Nagy PD.*
Title: *Kinetics and functional studies on interaction between the replicase proteins of Tomato Bushy Stunt Virus: Requirement of p33:p92 interaction for replicase assembly.*
PMID: *16242746.*

Heading	ACE
In vivo interaction between full length p33 and p92 replication proteins.	*perfect:* P33 interacts-with P92 in-vivo.
Kinetics of interaction between recombinant TBSV p33 replication proteins.	*not a fact.*
Role of highly conserved amino acid residues in S1 and S2 subdomains of p33 and p92 on the assembly of functional replication complexes and viral RNA replication in vivo.	*not a fact.*
TBSV p33 interacts with replicase proteins of the closely related CNV, but not with the more distantly related TCV.	*off-topic.*
Key role of the p33 replication co-factor in replicase assembly.	*off-topic.*
Model on the role of p33:p92 interaction during the assembly of the replicase complex.	*not a fact.*

Article #8

Authors: *Scott KL, Plon SE.*
Title: *CHES1/FOXN3 interacts with Ski-interacting protein and acts as a transcriptional repressor.*
PMID: *16102918.*

Heading	ACE
The C-terminus of CHES1 represses reporter transcription in human cells.	*off-topic.*
CHES1 interacts with SKIP.	*perfect:* CHES1 interacts-with SKIP.
Identification of the CHES1-binding domain on SKIP.	*not a fact.*

Article #9

AUTHORS: *Anand SP, Chattopadhyay A, Khan SA.*
TITLE: *The PcrA3 mutant binds DNA and interacts with the RepC initiator protein of plasmid pT181 but is defective in its DNA helicase and unwinding activities.*
PMID: *16122559.*

Heading	ACE
Overexpression and purification of the His-PcrA3 and MBP-RepC D57Y proteins.	*not a fact.*
The D57Y mutant of RepC is biologically active.	*off-topic.*
PcrA3 protein can interact with RepC.	*perfect:* PcrA3 interacts-with RepC.
DNA binding activity of PcrA3.	*off-topic.*
PcrA3 is defective in ATPase and DNA helicase activities.	*off-topic.*
The RepC D57Y mutant fails to complement PcrA3 in pT181 DNA unwinding and in vitro replication.	*off-topic.*

Article #10

AUTHORS: *Schiller MR, Blangy A, Huang J, Mains RE, Eipper BA.*
TITLE: *Induction of lamellipodia by Kalirin does not require its guanine nucleotide exchange factor activity.*
PMID: *15950621.*

Heading	ACE
Kal-GEF1, but not Kal-GEF2, induces formation of lamellipodia.	*off topic.*
Kalirin GEF1 activates Rac and Pak, proteins involved in forming lamellipodia.	*no match*[1].
Kalirin GEF1 induces lamellipodial formation through a GEF-activity-independent mechanism.	*off-topic.*
Induction of pinwheel lamellipodia by inactive Kal-GEF1 requires Pak kinase activity.	*off topic.*
Kal-GEF1 activation of Pak does not require GEF activity.	*no match*[2].
Kal-GEF1 interacts with Pak indirectly through Filamin A.	*perfect:* Kal-GEF1 indirectly interacts-with Pak. Kal-GEF1 directly interacts-with Filamin-A and Filamin-A directly interacts-with Pak.
A natural Kalirin isoform induces formation of lamellipodia and co-localizes with Pak.	*off topic. / perfect:* An isoform of Kalirin colocalizes-with Pak.
Lamellipodia induced by Kal-GEF1(ND/AA) share properties with lamellipodia induced by other mechanisms.	*off-topic.*

Article #11

AUTHORS: *Yao Q, Chen J, Cao H, Orth JD, McCaffery JM, Stan RV, McNiven MA.*
TITLE: *Caveolin-1 interacts directly with dynamin-2.*
PMID: *15811383.*

Heading	ACE
Dyn2 localizes to caveolae at the plasma membrane and interacts directly with Cav1.	*perfect:* Dyn2 localizes-to Caveola at Cell-Membrane and directly interacts-with Cav1.
The carboxy terminus including the PRD of Dyn2 mediates direct binding to Cav1.	*partial*[4]*:* The c-terminus of Dyn2 binds Cav1.
Cav1 interacts differentially with distinct Dyn2 forms.	*no match*[3].

Article #12

AUTHORS: *Zhang W, Arcos R.*
TITLE: *Interaction of the adenovirus major core protein precursor, pVII, with the viral DNA packaging machinery.*
PMID: *15780869.*

Heading	ACE
Interaction of pVII with the IVa2 protein	*perfect:* PVII interacts-with IVa2.
Interaction of pVII with the L1 52/55 kDa protein	*perfect:* PVII interacts-with L1-52-55-kDa.
Specific interaction of the IVa2 protein with the packaging sequence.	*off-topic.*
Interaction of the protein pVII with the packaging sequence.	*off-topic.*
Interaction of the L1 52/55 kDa protein with the packaging sequence.	*off-topic.*

Article #13

AUTHORS: *Xia L, Zheng L, Lee HW, Bates SE, Federico L, Shen B, O'Connor TR.*
TITLE: *Human 3-methyladenine-DNA glycosylase: effect of sequence context on excision, association with PCNA, and stimulation by AP endonuclease.*
PMID: *15713479.*

Heading	ACE
Both kcat and Km contribute to SCD MPG excision.	*partial*[3]*:* Kcat regulates the activity of SCD. Km regulates the activity of SCD.
Rate of MPG-catalyzed excision of Hx is enhanced in the presence of APE1.	*partial*[2]*:* APE1 increases the activity of MPG.
Enhancement of MPG-catalyzed SCD excision by APE1.	*partial (r)*[2]*:* APE1 increases the activity of MPG.
MPG associates with PCNA in vitro.	*perfect:* MPG associates-with PCNA in-vitro.
PCNA increases the rate of Hx excision by MPG.	*partial*[2]*:* PCNA increases the activity of MPG.
MPG excision of Hx is enhanced by PCNA on a nicked plasmid substrate with a unique damage site.	*partial (r)*[2]*:* PCNA increases the activity of MPG.
Co-immunoprecipitation of MPG, PCNA, and APE1 from human cells.	*not a fact.*

Article #14

AUTHORS: *Lilly Research Labs, Eli Lilly and Company, Lilly Corporate Center, Indianapolis, IN 46285, USA.*
TITLE: *Suramin interacts with RANK and inhibits RANKL-induced osteoclast differentiation.*
PMID: *15780954.*

Heading	ACE
Suramin inhibits osteoclast differentiation.	*off-topic.*
Suramin blocks PTH (138)-induced calvarial bone resorption.	*off-topic.*
Suramin blocks sRANKL-induced AKT and p38 MAP kinase phosphorylation.	*no match*[2]*.*
Suramin interferes with sRANKL binding to rhRANK-Fc.	*partial*[2]*:* SRANKL binds RhRANK-Fc.
Suramin binds to rhRANK-Fc	*perfect:* Suramin binds RhRANK-Fc.

Article #15

AUTHORS: *Torrado M, Nespereira B, Lopez E, Centeno A, Castro-Beiras A, Mikhailov AT.*
TITLE: *ANKRD1 specifically binds CASQ2 in heart extracts and both proteins are co-enriched in piglet cardiac Purkinje cells.*
PMID: *15698842.*

Heading	ACE
ANKRD1 specifically and selectively interacts with CASQ2 in cardiac extracts.	*perfect:* Ankrd1 specifically interacts-with CASQ2 in Cardiac-Extract and selectively interacts-with CASQ2 in Cardiac-Extract.
Direct reciprocal ANKRD1CASQ2 binding in vitro.	*partial*[1]*:* Ankrd1 directly binds CASQ2 in-vitro.
ANKRD1 contains potential CASQ2 binding sequences located in both its NT- and CT-regions.	*no match*[3]*.*
CASQ2 contains potential ANKRD1-binding sequences located outside its Asp-rich CT-region.	*no match*[3]*.*
ANKRD1 and CASQ2 are co-enriched in cardiac Purkinje cells.	*no match*[1]*.*

Article #16

AUTHORS: *Chun J, Kwon T, Lee EJ, Hyun S, Hong SK, Kang SS.*
TITLE: *The subcellular localization of 3-phosphoinositide-dependent protein kinase is controlled by caveolin-1 binding.*
PMID: *15567163.*

Heading	ACE
PDK1 interacts with caveolin-1 in vitro.	*perfect:* PDK1 interacts-with Caveolin-1 in-vitro.
PDK1 interacts with caveolin-1 in the COS-1 cell.	*perfect:* PDK1 interacts-with Caveolin-1 in COS-Cell.
The binding of caveolin-1 down-regulates both the self-phosphorylation and kinase activity of PDK1.	*partial*[1]*:* Caveolin-1 regulates the phosphorylation of PDK1 and regulates the activity of PDK1.
The caveolin-1 peptide also down-regulates both self-phosphorylation and kinase activity of PDK1 in vitro	*off-topic.*

Article #17

AUTHORS: *Schweneker M, Bachmann AS, Moelling K.*
TITLE: *The HIV-1 co-receptor CCR5 binds to alpha-catenin, a component of the cellular cytoskeleton.*
PMID: *15541354.*

Heading	ACE
The C-terminus of CCR5 interacts with a-catenin.	*perfect:* The c-terminus of CCR5 interacts-with Alpha-Catenin.
Association of a-catenin and mutants thereof with full-length CCR5 in mammalian cells.	*perfect:* Alpha-Catenin associates-with CCR5 in Mammal. A mutant of Alpha-Catenin associates-with CCR5 in Mammal.
Half-endogenous interaction of a-catenin with CCR5.	*partial (r)*[1]*:* CCR5 interacts-with Alpha-Catenin.
Endogenous interactions of a-catenin with the chemokine- and HIV-1 co-receptors CCR5 and CXCR4.	*partial*[1]*:* Alpha-Catenin interacts-with CCR5 and interacts-with CXCR4.

Article #18

AUTHORS: *Krause A, Zacharias W, Camarata T, Linkhart B, Law E, Lischke A, Miljan E, Simon HG.*
TITLE: *Tbx5 and Tbx4 transcription factors interact with a new chicken PDZ-LIM protein in limb and heart development.*
PMID: *15302601.*

Heading	ACE
Isolation and characterization of Tbx binding proteins.	*not a fact.*
Chicken LMP-4 is a novel member of the PDZ-LIM family of proteins.	*perfect:* Chicken-LMP-4 is a subtype of PDZ-LIM.
LMP-4 interacts specifically with the transactivation domains of Tbx5 and Tbx4, but not Tbx3.	*perfect:* LMP-4 specifically interacts-with a transactivation-domain of Tbx5 and specifically interacts-with a transactivation-domain of Tbx4 and does not specifically interact-with a transactivation-domain of Tbx3.
LMP-4 localizes Tbx5 and Tbx4 to actin filaments	*perfect:* LMP-4 localizes-to Microfilament. Tbx5 localizes-to Microfilament. Tbx4 localizes-to Microfilament.
LMP-4 mRNA in the developing limbs co-expresses with Tbx5 and Tbx4.	*off-topic.*
Tbx5 and Tbx4 expression domains in the developing heart co-localize with LMP-4 expression.	*off-topic.*

Article #19

AUTHORS: *Xu S, Hori RT.*
TITLE: *Identification of a domain within human TAF(I)48, a subunit of Selectivity Factor 1, that interacts with helix 2 of TBP.*
PMID: *15315821.*

Heading	ACE
Yeast two-hybrid strategy.	*not a fact.*
TBP binds more strongly to the carboxyl-terminal half of hTAFI48.	*partial*[3]*:* TBP binds the c-terminus of HTAF148.
The carboxyl-terminal 42 residues of hTAFI48 bind TBP.	*partial (r)*[1]*:* HTAFI148 binds TBP.
TBP binds directly to the carboxyl-terminus of hTAFI48	*perfect:* TBP directly binds the c-terminus of HTAF148.
TBP interacts with basic and polar residues within the carboxyl-terminus of hTAFI48.	*partial*[1]*:* TBP interacts-with a residue of the c-terminus of HTAF148.
The carboxyl-terminus of hTAFI48 interacts with residues within and around helix 2 of TBP.	*partial*[1]*:* The c-terminus of HTAF148 interacts-with a alpha-helix of TBP.

Article #20

Authors: *Johnson DR, Lovett JM, Hirsch M, Xia F, Chen JD.*
Title: *NuRD complex component Mi-2beta binds to and represses RORgamma-mediated transcriptional activation.*
PMID: *15144897.*

Heading	ACE
Identification of Mi-2beta as an RORgamma-interacting protein in yeast two-hybrid screen	*perfect:* Mi-2-Beta interacts-with ROR-Gamma in Yeast-Two-Hybrid.
Yeast two-hybrid interaction between Mi-2beta and RORgamma.	*perfect (r):* Mi-2-Beta interacts-with ROR-Gamma in Yeast-Two-Hybrid.
Mi-2beta interacts with RORgamma in vitro.	*perfect:* Mi-2-Beta interacts-with ROR-Gamma in-vitro.
Mi-2beta inhibits RORgamma transcriptional activity.	*partial*[1]*:* Mi-2-Beta inhibits the activity of ROR-Gamma.

Article #21

Authors: *Mizugishi K, Hatayama M, Tohmonda T, Ogawa M, Inoue T, Mikoshiba K, Aruga J.*
Title: *Myogenic repressor I-mfa interferes with the function of Zic family proteins.*
PMID: *15207726.*

Heading	ACE
Identification of the Zic2 transcriptional activation domain.	*not a fact.*
Zic family members specifically interact with I-mfa.	*perfect:* Zic specifically interacts-with I-mfa.
I-mfa inhibits transcriptional activities of Zic family proteins.	*partial*[1]*:* I-mfa inhibits the activity of Zic.
I-mfa can retain nuclear Zic proteins in the cytoplasm.	*off-topic.*

Article #22

Authors: *Wisniewski TP, Tanzi CL, Gindhart JG.*
Title: *The Drosophila kinesin-I associated protein YETI binds both kinesin subunits.*
PMID: *14720462.*

Heading	ACE
YETI binds specifically to both KHC and KLC in the yeast two-hybrid system.	*perfect:* YETI specifically binds KHC in Yeast-Two-Hybrid and specifically binds KLC in Yeast-Two-Hybrid.
YETI binds to kinesin in cultured Drosophila cells.	*perfect:* YETI binds Kinesin in Drosophila in Cultured-Cell.
YETI is localized to the nucleus and cytosol of cultured cells.	*perfect:* YETI localizes-to Nucleus. YETI localizes-to Cytosol in Cultured-Cell.

Article #23

Authors: *Prasad CK, Meyers C, Zhan DJ, You H, Chiriva-Internati M, Mehta JL, Liu Y, Hermonat PL.*
Title: *The adeno-associated virus major regulatory protein Rep78-c-Jun-DNA motif complex modulates AP-1 activity.*
PMID: *14517094.*

Heading	ACE
Rep78 binds c-Jun as demonstrated by Western blot analysis and amino half of Rep78 is required for this interaction.	*partial*[2]*:* Rep78 binds C-Jun in Western-Blot-Analysis.
Rep78 binds c-Jun in vivo by yeast two-hybrid cDNA analysis.	*perfect:* Rep78 binds C-Jun in-vivo in Yeast-Two-Hybrid-CDNA-Analysis.
Rep78 binds c-Jun in vitro as shown by EMSA supershift analysis.	*perfect:* Rep78 binds C-Jun in-vitro in EMSA-Supershift-Analysis.
Rep78 inhibits transcription from the c-jun promoter by CAT assay.	*off-topic.*
Rep78 inhibits c-Jun augmented transcription in vitro in nuclear extracts	*off-topic.*

Article #24

Authors: *Gieswein CE, Sharom FJ, Wildeman AG.*
Title: *Oligomerization of the E5 protein of human papillomavirus type 16 occurs through multiple hydrophobic regions.*
PMID: *12954209.*

Heading	ACE
HPV16 E5 can self-interact.	*perfect:* HPV16-E5 interacts-with itself.
HPV16 E5 monomers associate via hydrophobic interactions.	*perfect:* HPV16-E5 hydrophobically associates-with itself.
HPV16 E5 can interact with 16K through hydrophobic interactions.	*perfect:* HPV16-E5 hydrophobically interacts-with 16K.
Localization of HPV16 E5.	*not a fact.*

Article #25

Authors: *Shimoyama T, Kato K, Miyaji-Yamaguchi M, Nagata K.*
Title: *Synergistic action of MLL, a TRX protein with template activating factor-I, a histone chaperone.*
PMID: *15670842.*

Heading	ACE
In vivo interaction of MLLN with TAF-Ibeta	*perfect:* MLLN interacts-with TAF1B in-vivo.
In vitro interaction of MLLN with TAF-Ibeta	*perfect:* MLLN interacts-with TAF1B in-vitro.
Synergistic activation of Hoxa9 gene transcription by MLLN and TAF-Ibeta	*off-topic.*

Article #26

Authors: *Yamamoto K, Sonoda M.*
Title: *Self-interaction of heterochromatin protein 1 is required for direct binding to histone methyltransferase, SUV39H1.*
PMID: *12565857.*

Heading	ACE
Interaction between full-length mHP1alpha and full-length SUV39H1 in yeast and in vitro.	*perfect:* MHP1-Alpha interacts-with SUV39H1 in Yeast in-vitro.
Interaction sites of mHP1alpha and SUV39H1.	*not a fact.*
Self-interaction of mHP1alpha is required for binding to SUV39H1.	*partial*[2]*:* MHP1-Alpha interacts-with itself and binds SUV39H1.
Dimer surface of mHP1alpha is required for the interaction with SUV39H1.	*partial (r)*[1]*:* MHP1-Alpha interacts-with SUV39H1.

Article #27

Authors: *Chang JF, Hall BE, Tanny JC, Moazed D, Filman D, Ellenberger T.*
Title: *Structure of the coiled-coil dimerization motif of Sir4 and its interaction with Sir3.*
PMID: *12791253.*

Heading	ACE
Sir4 Dimerizes through a C-terminal Coiled-Coil Domain.	*partial*[1]*:* There is a dimer of Sir4. A c-terminal coiled-coil-domain of Sir4 binds itself.
Sir3 Binds to a Hydrophobic Patch on the Surface of the Sir4 Coiled Coil.	*partial*[1]*:* Sir3 binds a coiled-coil-domain of Sir4.
A Sir3 Dimer Binds to the Coiled Coil of Sir4.	*perfect:* A dimer of Sir3 binds a coiled-coil-domain of Sir4.
Assembly of a Ternary Complex of Sir2/Sir4/Sir3.	*perfect:* There is a protein-complex X such that Sir2 is a subunit of X and Sir4 is a subunit of X and Sir3 is a subunit of X.
Biological Implications.	*not a fact.:*

Article #28

Authors: *Cayrol C, Cougoule C, Wright M.*
Title: *The beta2-adaptin clathrin adaptor interacts with the mitotic checkpoint kinase BubR1.*
PMID: *12419313.*

Heading	ACE
Beta2 adaptin binds to BubR1 in yeast two-hybrid and in vitro binding assays.	*perfect:* Beta2-Adaptin binds BubR1 in Yeast-Two-Hybrid. Beta2-Adaptin binds BubR1 in Vitro-Binding-Assay.
The trunk domain of beta2-adaptin is necessary and sufficient for interaction with BubR1.	*partial*[1]*:* A trunk-domain of Beta2-Adaptin interacts-with BubR1.
Mapping of BubR1 domains involved in the binding to beta2-adaptin.	*not a fact.*
Immunolocalization studies of BubR1 and beta1/2 adaptins.	*not a fact.*
The BubR1 related mitotic checkpoint kinase Bub1 interacts with the trunk domain of beta2-adaptin.	*perfect:* Bub1 interacts-with the trunk-domain of Beta2-Adaptin.
The Bub1 and BubR1 kinases interact with all the beta chains of AP complexes.	*perfect:* Bub1 interacts-with every beta-sheet of AP. BubR1 interacts-with every beta-sheet of AP.

Article #29

Authors: *Du X, Hublitz P, Gunther T, Wilhelm D, Englert C, Schule R.*
Title: *The LIM-only coactivator FHL2 modulates WT1 transcriptional activity during gonadal differentiation.*
PMID: *12151099.*

Heading	ACE
FHL2 and WT1 are coexpressed in mouse embryonic gonads.	*off-topic.*
FHL2 interacts with WT1 in vitro.	*perfect:* FHL2 interacts-with WT1 in-vitro.
FHL2 interacts with WT1 in vivo.	*perfect:* FHL2 interacts-with WT1 in-vivo.
FHL2 modulates WT1-dependent transcription.	*off-topic.*

Article #30

Authors: *Payton JE, Perrin RJ, Clayton DF, George JM.*
Title: *Protein-protein interactions of alpha-synuclein in brain homogenates and transfected cells.*
PMID: *11687285.*

Heading	ACE
Tubulin co-immunoprecipitates with alpha-synuclein.	*partial*[1]*:* Tubulin interacts-with Alpha-Synuclein.
Tubulin binds immobilized GST/alpha-synuclein.	*partial*[1]*:* Tubulin binds GST-Alpha-Synuclein.
Alpha-Synuclein binds a tubulin affinity column.	*partial*[1]*:* Alpha-Synuclein binds Tubulin.
Alpha-Synuclein does not pellet with polymerized microtubules.	*off-topic.*
Alpha-Synuclein/GFP does not colocalize with microtubules in transfected cells.	*off-topic.*

Article #31

AUTHORS: *Kausalya PJ, Reichert M, Hunziker W.*
TITLE: *Connexin45 directly binds to ZO-1 and localizes to the tight junction region in epithelial MDCK cells.*
PMID: *11557048.*

Heading	ACE
The C-terminus of Cx45 interacts with the PDZ domains of ZO-1 and ZO-3 in a yeast two-hybrid assay.	*perfect:* The c-terminus of Cx45 interacts-with a PDZ-domain of ZO-1 in Yeast-Two-Hybrid. The c-terminus of Cx45 interacts-with a PDZ-domain of ZO-3 in Yeast-Two-Hybrid.
Characterization of epithelial MDCK cells transfected with Cx45 cDNA.	*not a fact.*
Cx45 directly associates with ZO-1 in vivo.	*perfect:* Cx45 directly associates-with ZO-1 in-vivo.
Cx45 co-localizes with ZO-1 in the tight junction region in polarized MDCK cells.	*perfect:* Cx45 colocalizes-with ZO-1 in Tight-Junction in Polarized-MDCK-Cell.

Article #32

AUTHORS: *Pellizzoni L, Baccon J, Charroux B, Dreyfuss G.*
TITLE: *The survival of motor neurons (SMN) protein interacts with the snoRNP proteins fibrillarin and GAR1.*
PMID: *11509230.*

Heading	ACE
SMN interacts directly with the snoRNP proteins fibrillarin and GAR1.	*perfect:* SMN directly interacts-with Fibrillarin and directly interacts-with GAR1.
The interaction of SMN with fibrillarin and GAR1 requires the conserved Y/G box and is defective in SMN mutants found in some SMA patients.	*partial (r)*[4]*:* SMN interacts-with Fibrillarin and interacts-with GAR1.
The arginine- and glycine-rich domains of fibrillarin and GAR1 are necessary for SMN interaction.	*no match*[1]*.*
Association of the SMN complex with fibrillarin and GAR1 in vivo.	*perfect:* SMN associates-with Fibrillarin in-vivo and associates-with GAR1 in-vivo.
Transcription-dependent association of SMN with fibrillarin, GAR1, and the nucleolus.	*partial*[1]*:* SMN associates-with Fibrillarin and associates-with GAR1.
Expression of SMNdeltaN27 causes accumulation of snoRNPs outside the nucleolus.	*off-topic.*

Article #33

AUTHORS: *Jones DD, Stott KM, Reche PA, Perham RN.*
TITLE: *Recognition of the lipoyl domain is the ultimate determinant of substrate channelling in the pyruvate dehydrogenase multienzyme complex.*
PMID: *11114246.*

Heading	ACE
Interaction of E2plipapo with E1p.	*perfect:* E2plipapo interacts-with E1p.
The effect of 2-oxo acid on the interaction of E2plipapo with E1p.	*not a fact.*
The interaction of E2plipholo with E1p.	*perfect:* E2plipholo interacts-with E1p.
The interaction of E2plipholo with E1o and BSA.	*perfect:* E2plipholo interacts-with E1o and interacts-with BSA.
Mapping the interaction sites on the surface of E2plip.	*not a fact.*

Article #34

AUTHORS: *Shapiro R, Ruiz-Gutierrez M, Chen CZ.*
TITLE: *Analysis of the interactions of human ribonuclease inhibitor with angiogenin and ribonuclease A by mutagenesis: importance of inhibitor residues inside versus outside the C-terminal "hot spot".*
PMID: *10970748.*

Heading	ACE
Role of the C-terminal region of hRI.	*not a fact.*
Interactions of des(460)-hRI with the Ang variants R5A and K40G.	*partial*[1]*:* Des-460-hRI interacts-with a mutant of ANG.
Interactions of hRI variants Q430A/V432A, W438A/S439A/E440A, R457A, and I459A with Ang and RNase A.	*partial*[1]*:* A mutant of HRI interacts-with ANG and interacts-with RNase-A. A mutant of HRI interacts-with ANG and interacts-with RNase-A. A mutant of HRI interacts-with ANG and interacts-with RNase-A. A mutant of HRI interacts-with ANG and interacts-with RNase-A.
Role of the tryptophan-rich region of hRI.	*not a fact.*
Effects of single-residue Ang and hRI replacements on hRI-Ang binding affinity.	*not a fact.*
Interactions of multi-residue hRI variants with Ang.	*not a fact.*
Effects of hRI replacements on affinity for RNase A.	*not a fact.*
Inter-relationship between the Trp-rich and hot spot regions in the hRI-Ang complex.	*not a fact.*
Role of hRI Tyr150 in binding Ang.	*not a fact.*

Article #35

Authors: *Rotheneder H, Geymayer S, Haidweger E.*
Title: *Transcription factors of the Sp1 family: interaction with E2F and regulation of the murine thymidine kinase promoter.*
PMID: *10547281.*

Heading	ACE
Amino acids 102-125 of E2F-1 and 622-668 of Sp1 are sufficient for interaction of the two proteins.	*partial*[1]*:* E2F1 interacts-with Sp1.
Cyclin A does not interfere with the binding of Sp1 to E2F-1.	*no match*[2]*.*
All members of the Sp1 family are able to interact with E2F-1.	*perfect:* Every protein that is a subtype of Sp1 interacts-with E2F1.
Sp1 and Sp3 bind to the Sp1 binding site of the mouse TK promoter in electrophoretic mobility shift assays.	*perfect:* Sp1 binds a region X of Mouse-TK-Promoter in Electrophoretic-Mobility-Shift-Assay and Sp3 binds the region X of Mouse-TK-Promoter in Electrophoretic-Mobility-Shift-Assay.
Sp1, Sp3 and to a lesser extent Sp4, but not Sp2 are able to transactivate the mouse TK promoter in SL2 cells.	*off-topic.*
The TK promoter is synergistically activated in mouse fibroblasts.	*off-topic.*

Article #36

Authors: *Ayora S, Stasiak A, Alonso JC.*
Title: *The Bacillus subtilis bacteriophage SPP1 G39P delivers and activates the G40P DNA helicase upon interacting with the G38P-bound replication origin.*
PMID: *10329127.*

Heading	ACE
Purification of SPP1 G39P.	*not a fact.*
G39P specifically interacts with G40P.	*perfect:* G39P specifically interacts-with G40P.
Stoichiometry of G40P-G39P interaction.	*not a fact.*
G39P interferes with the high affinity binding of G40P to ssDNA.	*no match*[2]*.*
G39P decreases the ATPase activity of G40P.	*partial*[1]*:* G39P decreases the activity of G40P.
G38P reverses the negative effect exerted by G39P on the DNA-dependent ATPase activity of G40P.	*no match*[2]*.*
The G38P-G39P complex increases the apparent unwinding activity of G40P.	*partial*[1]*:* There is a protein-complex X such that G38P is a subunit of the protein-complex X and G39P is a subunit of the protein-complex X and the protein-complex X increases the activity of G40P.
Release of G39P upon formation of an oriL-G38P-G39P-G40P-ATP?S complex.	*no match*[4]*.*

Article #37

AUTHORS: *Wunderlich L, Goher A, Farago A, Downward J, Buday L.*
TITLE: *Requirement of multiple SH3 domains of Nck for ligand binding.*
PMID: *10372803.*

Heading	ACE
Nck Associates with Bcr/Abl, Cbl, p120 and p155 In Vivo.	*perfect:* Nck associates-with Bcr-Abl in-vivo and associates-with Cbl in-vivo and associates-with P120 in-vivo and associates-with P155 in-vivo.
Nck Interacts with Bcr/Abl, Cbl and p155 via its SH3 Domains.	*perfect:* A SH3-domain of Nck interacts-with Bcr-Abl and interacts-with Cbl and interacts-with P155.
Combined SH3 Domains of Nck are Necessary for Ligand Binding.	*off-topic.*
Mutations of Conserved Tryptophans to Lysines in GST-SH3-12 Abolishes Ligand Binding.	*off-topic.*

Article #38

AUTHORS: *Tanimura S, Ohtsuka S, Mitsui K, Shirouzu K, Yoshimura A, Ohtsubo M.*
TITLE: *MDM2 interacts with MDMX through their RING finger domains.*
PMID: *10218570.*

Heading	ACE
Identification of MDMX as a MDM2-binding protein.	*perfect:* MDMX binds MDM2.
Binding of MDMX with MDM2 in vitro and in vivo.	*perfect:* MDMX binds MDM2 in-vitro and binds MDM2 in-vivo.
Effect of co-expression of MDMX on MDM2 stability.	*off-topic.*

Article #39

AUTHORS: *Miyata KS, McCaw SE, Meertens LM, Patel HV, Rachubinski RA, Capone JP.*
TITLE: *Receptor-interacting protein 140 interacts with and inhibits transactivation by, peroxisome proliferator-activated receptor alpha and liver-X-receptor alpha.*
PMID: *10022764.*

Heading	ACE
Interaction cloning of RIP140.	*not a fact.*
RIP140 interacts with PPARalpha, RXRalpha and LXRalpha in vitro.	*perfect:* PIP140 interacts-with PPAR-Alpha in-vitro and interacts-with RXR-Alpha in-vitro and interacts-with LXR-Alpha in-vitro.
RIP140 antagonizes PPARalpha/RXRalpha- and LXRalpha/RXRalpha -mediated signaling.	*off-topic.*

Article #40

AUTHORS: *Ng RW, Arooz T, Yam CH, Chan IW, Lau AW, Poon RY.*
TITLE: *Characterization of the cullin and F-box protein partner Skp1.*
PMID: *9827542.*

Heading	ACE
Interaction between Skp1 and the F-box-containing protein Skp2.	*perfect:* Skp1 interacts-with Skp2.
Expression and purification of recombinant Skp1.	*not a fact.*
The native molecular size of Skp1.	*not a fact.*
Skp1 associates with Skp2 as well as other proteins in mammalian cell extracts.	*partial*[3]*:* Skp1 associates-with Skp2 in Mammal.

Article #41

AUTHORS: *Woo HN, Hong GS, Jun JI, Cho DH, Choi HW, Lee HJ, Chung CW, Kim IK, Jo DG, Pyo JO, Bertin J, Jung YK.*
TITLE: *Inhibition of Bcl10-mediated activation of NF-kappa B by BinCARD, a Bcl10-interacting CARD protein.*
PMID: *15637807.*

Heading	ACE
Identification and expression of BinCARD.	*not a fact.*
BinCARD binds to Bcl10 through CARD–CARD interaction.	*perfect:* A CARD-domain of BinCARD binds a CARD-domain of BCL10.
BinCARD inhibits Bcl10-mediated activation of NF-kappaB.	*partial*^{2}*:* BinCARD inhibits the activity of NF-kappaB.
BinCARD reduces the phosphorylation of Bcl10.	*perfect:* BinCARD decreases the phosphorylation of BCL10.
BinCARD inhibits Bcl10 phosphorylation induced by T cell activation signal.	*partial*^{2}*:* BinCARD inhibits the phosphorylation of BCL10.

Article #42

Authors: *Yoshima T, Yura T, Yanagi H.*
Title: *Novel testis-specific protein that interacts with heat shock factor 2.*
PMID: *9651507.*

Heading	ACE
cDNA cloning of Image.	*not a fact.*
Testis-specific expression of Image.	*not a fact.*
Interaction of HSF2 with HSF2BP in vitro.	*perfect:* HSF2 interacts-with HSF2BP in-vitro.
Two-hybrid assays for HSF2BP–HSF2 interaction in mammalian cells.	*perfect:* HSF2BP interacts-with HSF2 in Two-Hybrid-Assay in Mammal.

Article #43

Authors: *Ibarrola I, Vossebeld PJ, Homburg CH, Thelen M, Roos D, Verhoeven AJ.*
Title: *Influence of tyrosine phosphorylation on protein interaction with FcgammaRIIa.*
PMID: *9268059.*

Heading	ACE
FcgammaRIIa phosphorylation by various tyrosine kinases.	*perfect:* Tyrosine-Kinase phosphorylates FcGammaRIIa.
Lyn association with FcgammaRIIa.	*perfect:* Lyn associates-with FcGammaRIIa.
Interaction of cytosolic proteins with phosphorylated GST-CT.	*partial*^{1}*:* Cytosolic-Protein interacts-with GST-CT.
Differential roles of tyrosine residues in FcgammaRIIa.	*not a fact.*

Article #44

Authors: *Choi YJ, Cho SY, Kim HW, Kim JA, Bae SH, Park SS.*
Title: *Cloning and characterization of mouse disabled 2 interacting protein 2, a mouse orthologue of human NOSTRIN.*
PMID: *15596140.*

Heading	ACE
Yeast-two hybrid screening.	*not a fact.*
Nucleotide and deduced amino acid sequences of mDaIP2.	*not a fact.*
Expression pattern of mDaIP2 in transcriptional and translational level.	*not a fact.*
mDaIP2 binds to the mDab2 in F9 cells treated with RA.	*partial*^{1}*:* MDaIP2 binds MDab2 in F9-Cell.
Subcellular localization of mDaIP2.	*not a fact.*

Article #45

Authors: *Afshar K, Willard FS, Colombo K, Johnston CA, McCudden CR, Siderovski DP, Gonczy P.*
Title: *RIC-8 is required for GPR-1/2-dependent Galpha function during asymmetric division of C. elegans embryos.*
PMID: *15479639.*

Heading	ACE
ric-8 Is Required for Proper Asymmetric Division of C. elegans Embryos.	*off-topic.*
RIC-8 Distribution.	*not a fact.*
ric-8 Is Required for Generation of Pulling Forces on Spindle Poles.	*off-topic.*
RIC-8 Interacts with GOA-1 and GPA-16.	*perfect:* RIC-8 interacts-with Goa-1 and interacts-with GPA-16.
RIC-8 Is a GEF and GPR-1/2 Is a GDI.	*perfect:* RIC-8 is a subtype of GEF. GPR-1-2 is a subtype of GDI.
RIC-8 Is Required for Interaction of GOA-1 with GPR-1/2.	*partial*[1]*:* Goa-1 interacts-with GPR-1-2.
Inactivation of Gbetagamma Alleviates the Requirement for RIC-8 in Asymmetric Cell Division.	*no match*[2]*.*

Article #46

Authors: *Tai HH, Geisterfer M, Bell JC, Moniwa M, Davie JR, Boucher L, McBurney MW.*
Title: *CHD1 associates with NCoR and histone deacetylase as well as with RNA splicing proteins.*
PMID: *12890497.*

Heading	ACE
CHD1 associates with HDAC.	*perfect:* CHD1 associates-with HDAC.
CHD1 interacts with NCoR and splicing proteins.	*perfect:* CHD1 interacts-with NCoR and interacts-with Splicing-Protein.
In vitro pull-down assay confirms interaction between CHD1 and NcoR.	*not a fact.*
CHD1 and CLK1 interact with different regions of NcoR.	*perfect:* CHD1 interacts-with a region X of NCoR and CLK1 interacts-with a region Y of NCoR and the region X is not the region Y.
CHD1 and splicing.	*not a fact.*

Article #47

Authors: *Tsuzuki M, Wu W, Nishikawa H, Hayami R, Oyake D, Yabuki Y, Fukuda M, Ohta T.*
Title: *A truncated splice variant of human BARD1 that lacks the RING finger and ankyrin repeats.*
PMID: *15878232.*

Heading	ACE
Identification of BARD1deltaRIN.	*not a fact.*
DeltaRIN expression in human cell lines.	*not a fact.*
DeltaRIN localizes to the cytoplasm and does not interact with BRCA1.	*perfect:* Delta-RIN localizes-to Cytoplasm and does not interact-with BRCA1.
DeltaRIN interacts with and colocalizes with CstF-50.	*perfect:* Delta-RIN interacts-with CstF-50 and colocalizes-with CstF-50.

Article #48

Authors: *Yan J, Zhu J, Zhong H, Lu Q, Huang C, Ye Q.*
Title: *BRCA1 interacts with FHL2 and enhances FHL2 transactivation function.*
PMID: *14550570.*

Heading	ACE
Identification of a BRCA1-interacting protein.	*not a fact.*
Interaction between FHL2 and BRCA1 in vitro and in vivo.	*perfect:* FHL2 interacts-with BRCA1 in-vitro and interacts-with BRCA1 in-vivo.
Mapping of the BRCA1 binding domain of FHL2.	*not a fact.*
Potentiation of the FHL2 transactivation by BRCA1.	*no match*[4]*.*
Lack of BRCA1 binding sites in the FHL2 abolishes FHL2 transactivation function.	*no match*[4]*.*
Expression of FHL2 mRNA in breast cancer cell lines.	*not a fact.*

Article #49

AUTHORS: *Mils V, Lee SM, Joly W, Hang EW, Baldin V, Waye MM, Ducommun B, Tsui SK.*
TITLE: *LIM-only protein FHL3 interacts with CDC25B2 phosphatase.*
PMID: *12681290.*

Heading	ACE
Interaction between FHL3 and the CDC25B2 phosphatase in a two-hybrid assay.	*perfect:* FHL3 interacts-with CDC25B2 in Two-Hybrid-Assay.
In vitro interaction assay	*not a fact.*
In vivo FHL3 and CDC25B2 interaction is limited to the nucleus.	*perfect:* If FHL3 interacts-with CDC25B2 in-vivo in a cellular-component X then X is a part of Nucleus.
FHL3 has no effect on CDC25B2 phosphatase activity.	*no match*[1].

Article #50

AUTHORS: *Backes WL, Kelley RW.*
TITLE: *Organization of multiple cytochrome P450s with NADPH-cytochrome P450 reductase in membranes.*
PMID: *12725870.*

Heading	ACE
Evidence for specific interactions between different P450 enzymes.	*perfect:* P450 specifically interacts-with itself.
Demonstration that interactions between CYP2B4 and CYP1A2 occur in microsomes.	*perfect:* CYP2B4 interacts-with CYP1A2 in Microsome.
Possible interactions among multiple P450s and reductase.	*not a fact.*

Article #51

AUTHORS: *Lin CL, Leu S, Lu MC, Ouyang P.*
TITLE: *Over-expression of SR-cyclophilin, an interaction partner of nuclear pinin, releases SR family splicing factors from nuclear speckles.*
PMID: *15358154.*

Heading	ACE
Identification of SR-cyp as an interacting partner of pnn.	*perfect:* SR-cyp interacts-with PNN.
Interaction between SR-cyp and pnn is mediated by pnns C-terminal RS domain.	*perfect:* SR-cyp interacts-with the c-terminal RS-domain of PNN.
SR-cyp regulates SR family proteins intranuclear distribution by releasing them from speckles to nucleoplasm.	*no match*[4].

Article #52

AUTHORS: *Joensen L, Borda E, Kohout T, Perry S, Garcia G, Sterin-Borda L.*
TITLE: *Trypanosoma cruzi antigen that interacts with the beta1-adrenergic receptor and modifies myocardial contractile activity.*
PMID: *12672526.*

Heading	ACE
Recombinant MBP-Tc13 Tul interacts with the beta1-AR.	*partial*[1]*:* MBP-Tc13 interacts-with Beta1-AR.
Effects of MBP-Tc13 Tul on atrial function.	*not a fact.*

Article #53

AUTHORS: *Ramm K, Pluckthun A.*
TITLE: *High enzymatic activity and chaperone function are mechanistically related features of the dimeric E. coli peptidyl-prolyl-isomerase FkpA.*
PMID: *11428902.*

Heading	ACE
High isomerase activity with protein substrates mediated by tight substrate binding.	*off-topic.*
FK520 binds with nanomolar affinity to both active sites of a dimeric FkpA.	*partial*[1]*:* FK520 binds a dimer of FkpA.
RCM a-lactalbumin competes for protein substrate binding only.	*off-topic.*
FkpA prevents aggregation of unfolding CS intermediates.	*no match*[4]*.*
FkpA binds reversibly to early unfolding intermediates of CS.	*no match*[4]*.*
Binding of CS-unfolding intermediates is not influenced by inhibition of isomerase activity.	*no match*[4]*.*
Both activities reside primarily in the FKBP-domain.	*no match*[4]*.*
FkpY from Haemophilus influenzae has identical characteristics.	*no match*[4]*.*

Article #54

Authors: *Ji YJ, Nam S, Jin YH, Cha EJ, Lee KS, Choi KY, Song HO, Lee J, Bae SC, Ahnn J.*
Title: *RNT-1, the C. elegans homologue of mammalian RUNX transcription factors, regulates body size and male tail development.*
PMID: *15385167.*

Heading	ACE
The rnt-1(ok351) mutation bears a deletion.	*off-topic.*
The rnt-1(ok351) mutant has a small body size and male tail defects.	*off-topic.*
rnt-1 is expressed in the male tail.	*off-topic.*
rnt-1 is expressed in the male tail.	*off-topic.*
The rnt-1(ok351) mutant shows synergistic effects with known mutants of Sma/Mab pathway genes.	*off-topic.*
The phenotype of lon-1(e185), which is the downstream target mutant of Sma/Mab pathway, is epistatic to the rnt-1(ok351) mutant phenotype.	*off-topic.*
RNT-1 physically interacts with SMA-4.	*perfect:* RNT-1 physically interacts-with SMA-4.

Article #55

Authors: *Nishanian TG, Waldman T.*
Title: *Interaction of the BMPR-IA tumor suppressor with a developmentally relevant splicing factor.*
PMID: *15351706.*

Heading	ACE
BMPR-IA yeast two-hybrid screening.	*not a fact.*
Full-length BMPR-IA interacts with SAP49 in human cells.	*perfect:* BMPR1A interacts-with SAP49 in Human.
BMPR-IA and SAP49 localization.	*not a fact.*
Characterization of BMPR-IA/SAP49 interaction.	*not a fact.*
SAP49 expression analysis.	*not a fact.*
Functional consequence of BMPR-IA/SAP49 interaction.	*not a fact.*
SAP49 is a developmentally relevant splicing factor.	*off-topic.*
Interaction between splicing components and cell signaling.	*off-topic.*
Effects of BMPR-IA on SAP49 function.	*not a fact.*

Article #56

Authors: *Yanagita M, Oka M, Watabe T, Iguchi H, Niida A, Takahashi S, Akiyama T, Miyazono K, Yanagisawa M, Sakurai T.*
Title: *USAG-1: a bone morphogenetic protein antagonist abundantly expressed in the kidney.*
PMID: *15020244.*

Heading	ACE
Primary structure of USAG-1.	*not a fact.*
Secreted form of human USAG-1.	*not a fact.*
USAG-1 antagonizes the action of BMPs in C2C12 cells.	*perfect:* USAG-1 decreases the activity of BMP in C2C12-Cell.
USAG-1 inhibits endogenous BMP activity in Xenopus embryogenesis.	*perfect:* USAG-1 inhibits the activity of BMP in Xenopus in Embryogenesis.
USAG-1 directly binds BMP-2, -4, and -7.	*perfect:* USAG-1 directly binds BMP2 and directly binds BMP4 and directly binds BMP7.
Tissue distribution of USAG-1 in fetal and adult mice.	*not a fact.*
Effects of USAG-1 on Wnt1 signaling.	*not a fact.*

Article #57

AUTHORS: *Yoshida Y, von Bubnoff A, Ikematsu N, Blitz IL, Tsuzuku JK, Yoshida EH, Umemori H, Miyazono K, Yamamoto T, Cho KW.*
TITLE: *Tob proteins enhance inhibitory Smad-receptor interactions to repress BMP signaling.*
PMID: *12782279.*

Heading	ACE
Tob and Tob2 preferentially interact with inhibitory Smads.	*partial*[3]*:* Tob interacts-with Smad. Tob2 interacts-with Smad.
Identification and expression of Xenopus Tob2.	*not a fact.*
Both Tob and XTob2 cooperate with Smad6 to inhibit BMP signaling.	*off-topic.*
Tob co-localizes with Smad6 at the cell membrane and assists in complex formation between inhibitory Smads and BMP receptors.	*partial*[2]*:* Tob colocalizes-with SMAD6 at Cell-Membrane. There is a protein-complex X such that Smad is a subunit of X and a receptor of BMP is a subunit of X.

Article #58

AUTHORS: *Kraemer B, Crittenden S, Gallegos M, Moulder G, Barstead R, Kimble J, Wickens M.*
TITLE: *NANOS-3 and FBF proteins physically interact to control the sperm-oocyte switch in Caenorhabditis elegans.*
PMID: *10508609.*

Heading	ACE
NOS-3 and FBF proteins interact physically.	*perfect:* Nos-3 physically interacts-with FBF.
The nos-3 mRNA and protein are present in the germ line and present throughout development.	*off-topic.*
NOS functions in the hermaphrodite spermoocyte switch.	*off-topic.*
NOS functions in the hermaphrodite spermoocyte switch.	*off-topic.*
The three nos genes are critical for germ-line survival.	*off-topic.*
Overlapping but non-identical functions of the nos genes.	*off-topic.*
Other nos defects.	*not a fact.*

Article #59

AUTHORS: *Upton JW, van Dyk LF, Speck SH.*
TITLE: *Characterization of murine gammaherpesvirus 68 v-cyclin interactions with cellular cdks.*
PMID: *16102793.*

Heading	ACE
GammaHV68 v-cyclin interacts with cdk2 and cdc2, but not cdk4 or cdk6.	*perfect:* GammaHV68-V-Cyclin interacts-with Cdk2 and interacts-with Cdc2 and does not interact-with Cdk4 and does not interact-with Cdk6.
Mutation of conserved residues within the cyclin box of gammaHV68 v-cyclin eliminates binding to cellular cdks.	*no match*[1]*.*
GammaHV68 v-cyclin:cdk2 complexes phosphorylate cyclin E:cdk2 substrates.	*off-topic.*
GammaHV68 v-cyclin-mediated phosphorylation of cyclin A:cdk substrates.	*off-topic.*

Article #60

AUTHORS: *Nevado J, Tenbaum SP, Aranda A.*
TITLE: *hSrb7, an essential human Mediator component, acts as a coactivator for the thyroid hormone receptor.*
PMID: *15249124.*

Heading	ACE
hSrb7 and hMo15 interact with the thyroid hormone receptor in mammalian two-hybrid assays.	*perfect:* HSrb7 interacts-with Thyroid-Hormone-Receptor in Mammalian-Two-Hybrid-Assay. HMo15 interacts-with Thyroid-Hormone-Receptor in Mammalian-Two-Hybrid-Assay.
Srb7 enhances TR-mediated transcriptional activation.	*off-topic.*
Transcriptional synergistic effect between hSrb7 and TR is T3- and AF-2-dependent.	*off-topic.*
Gal4DBD-fused ligand binding domains of TRa and TR also show a synergistic effect with hSrb7, and other basal-transcriptional components.	*no match*[4].
TR interacts directly with hSrb7 in vitro.	*perfect:* TR directly interacts-with HSrb7 in-vitro.
hSrb7 does not interact with other nuclear receptors.	*no match*[4].
hSrb7 does not act as a coactivator for other nuclear receptors.	*no match*[4].

Article #61

AUTHORS: *Cordenonsi M, Dupont S, Maretto S, Insinga A, Imbriano C, Piccolo S.*
TITLE: *Links between tumor suppressors: p53 is required for TGF-beta gene responses by cooperating with Smads.*
PMID: *12732139.*

Heading	ACE
Cloning of an Alternatively Spliced Isoform of p53 (p53AS) in a Screen for Activators of TGF-beta Signaling.	*not a fact.*
p53 Is Required for TGF-beta/Activin/Nodal-Mediated Gene Responses in Xenopus Embryos.	*off-topic.*
p53 Is Required for Full TGF-beta Gene Responses and TGF-beta-Mediated Growth Arrest in Mammalian Cells.	*off-topic.*
p53 Is Required for TGF-beta1-Mediated Growth Arrest in Mouse Embryonic Fibroblasts and Hematopoietic Progenitors.	*off-topic.*
p53 Family Members Cooperate with TGF-beta-Induced Transcription in Human Cells but Require Promoter Sequences Separate from Smad-Responsive Element.	*off-topic.*
A p53 Responsive Element on the Mix.2 Promoter Is Required for Endogenous Expression In Vivo and Full TGF-beta Responsiveness.	*off-topic.*
p53 Physically Interacts with Smads.	*perfect:* P53 physically interacts-with Smad.

Article #62

AUTHORS: *Rascle A, Stowers RS, Garza D, Lepesant JA, Hogness DS.*
TITLE: *L63, the Drosophila PFTAIRE, interacts with two novel proteins unrelated to cyclins.*
PMID: *12782278.*

Heading	ACE
Two Drosophila genes encode proteins that interact with the L63B1 isoform.	*off-topic.*
PIF-1B and PIF-2 interact with the same histidine-rich domain present in the N-terminal extension of all L63 isoforms.	*perfect:* There is a histine-rich-domain X that is a part of the n-terminus of every isoform of L63 and PIF-1B interacts-with X and PIF-2 interacts-with X.
L63B1 interacts with cysteine-rich domains in PIF-1B and PIF-2.	*perfect:* L63B1 interacts-with a cysteine-rich-domain of PIF-1B. L63B1 interacts-with a cysteine-rich-domain of PIF-2.
Developmental profiles of PIF-1 mRNA and protein abundances indicate that PIF-1 expression is controlled at both transcriptional and translational levels.	*off-topic.*
The domain in PIF-1B that is required for interaction with L63 in yeast binds L63 in vitro.	*perfect:* A domain X of PIF-1B interacts-with L63 in Yeast and the domain X binds L63 in-vitro.
L63 is an active kinase.	*perfect:* L63 is a subtype of Active-Kinase.
Coimmunoprecipitation assays indicate that L63 and PIF-1B interact in vivo.	*perfect:* L63 interacts-with PIF-1B in-vivo in Coimmunoprecipitation-Assay.

Article #63

Authors: *Vogel L, Baratte B, Detivaud L, Azzi L, Leopold P, Meijer L.*
Title: *Molecular cloning and characterisation of p15(CDK-BP), a novel CDK-binding protein.*
PMID: *12007796.*

Heading	ACE
cDNA cloning of p15CDK-BP.	*not a fact.*
Starfish suc1 homologue cloning and sequence analysis of the p15A and p15B with the p13suc1/p9Cks family of proteins.	*not a fact.*
p15CDK-BP transcripts are restricted to oocytes.	*off-topic.*
Recombinant p15B possesses the same properties as purified p15CDK-BP and binds a starfish PSTAIRE-immunoreactive protein.	*no match*[4].
p15CDK-BP binds several CDKs.	*perfect:* P15CDK-BP binds CDK.

Article #64

Authors: *Haidweger E, Novy M, Rotheneder H.*
Title: *Modulation of Sp1 activity by a cyclin A/CDK complex.*
PMID: *11237594.*

Heading	ACE
Sp1 interacts with a growth dependent histone 1 (H1) kinase activity.	*no match*[4].
Cyclin A interacts with Sp1.	*perfect:* Cyclin-A interacts-with Sp1.
The zinc finger region of Sp1 and the N terminus of cyclin A are necessary for the interaction of the two proteins.	*no match*[4].
A cyclin A/kinase complex is able to phosphorylate Sp1 and to enhance its DNA binding activity.	*no match*[4].
nhibition of CDK activity reduces DNA binding of Sp1 and Sp1 dependent expression of a reporter gene.	*off-topic.*
Over-expression of cyclin A increases DNA binding of Sp1 and enhances Sp1 dependent promoter activity.	*off-topic.*
Stimulation of arrested 3T6 cells to grow results in enhanced DNA binding of Sp1/Sp3 which correlates with the expression of cyclin A.	*off-topic.*

Article #65

Authors: *Wang Y, Xu F, Hall FL.*
Title: *The MAT1 cyclin-dependent kinase-activating kinase (CAK) assembly/targeting factor interacts physically with the MCM7 DNA licensing factor.*
PMID: *11056214.*

Heading	ACE
Identification of MCM7 as a putative MAT1-interacting protein.	*no match*[3].
Verification of the in vivo interaction between MAT1 and MCM7 in the yeast two-hybrid system.	*not a fact.*
Verification of the physical interaction between MAT1 and MCM7 in vitro.	*not a fact.*
Association of MCM7 with CAK complexes (CDK7, cyclin H, MAT1) in mammalian cells.	*perfect:* MCM7 associates-with CDK7 in Mammal and associates-with Cyclin-H in Mammal and associates-with MAT1 in Mammal.

Article #66

AUTHORS: *Moorthamer M, Zumstein-Mecker S, Chaudhuri B.*
TITLE: *DNA binding protein dbpA binds Cdk5 and inhibits its activity.*
PMID: *10100871.*

Heading	ACE
A C-terminal fragment of dbpA binds to Cdk5 in a yeast two-hybrid screen.	*perfect:* A c-terminal region of DbpA binds Cdk5 in Yeast-Two-Hybrid.
The dbpA(Cdelta) protein precipitates 35S-labeled Cdk5 and Cdk4.	*no match*[4].
The dbpA(Cdelta) protein precipitates Cdk5 and Cdk4 expressed in COS-1 cells.	*off-topic.*
issue specific expression of dbpA.	*off-topic.*
Inhibition of the Cdk5 kinase by bacterially expressed GST-dbpA(Cdelta) and His-dbpA(Cdelta).	*no match*[4].
Activation of Cdk5 with D-type cyclins.	*no match*[4].
Inhibition of the Cdk4 kinase by bacterially expressed His-dbpA(Cdelta).	*no match*[4].
Specificity of inhibition of the Cdk5 kinase compared to the Cdk2 kinase.	*not a fact.*
Triple infections with GST-DbpA(C?) baculoviruses do not yield active Cdk5 or Cdk4 kinases.	*off-topic.*
Inhibition of the Cdk5 and Cdk4 kinase by insect cell expressed GST-DbpA(Cdelta).	*no match*[4].

Article #67

AUTHORS: *Tran DD, Edgar CE, Heckman KL, Sutor SL, Huntoon CJ, van Deursen J, McKean DL, Bram RJ.*
TITLE: *CAML is a p56Lck-interacting protein that is required for thymocyte development.*
PMID: *16111633.*

Heading	ACE
Creation of Conditional CAML Knockout Mice in Thymocytes.	*not a fact.*
Reduced Numbers of DP and SP Thymocytes in tCAML–/– Mice.	*off-topic.*
Thymocytes Lacking CAML Do Not Contribute to the Peripheral T Cell Subset.	*off-topic.*
Positive Selection Is Impaired, Whereas Negative Selection Is Enhanced in tCAML–/– Thymocytes.	*off-topic.*
CAML-Deficient Thymocytes Undergo Increased Cell Death upon TCR Ligation.	*off-topic.*
CAML Interacts with p56Lck in an Activation-Dependent Manner.	*partial*[1]*:* CAML interacts-with P56Lck.
CAML Negatively Regulates p56Lck Activation.	*partial*[1]*:* CAML regulates the activity of P56Lck.

Article #68

AUTHORS: *Warbrick E, Lane DP, Glover DM, Cox LS.*
TITLE: *A small peptide inhibitor of DNA replication defines the site of interaction between the cyclin-dependent kinase inhibitor p21WAF1 and proliferating cell nuclear antigen.*
PMID: *7780738.*

Heading	ACE
The carboxy-terminal 89 amino acids of p21WAF1 bind PCNA in an interaction trap.	*partial*[1]*:* A c-terminal region of P21WAF1 binds PCNA.
PCNA–PCNA interactions.	*perfect:* PCNA interacts-with itself.
p21WAF1 interacts with the central region of PCNA.	*perfect:* P21WAF1 interacts-with the central-region of PCNA.
Peptide mapping of sites on p21WAF1 important for its interaction with PCNA.	*off-topic.*
p21WAF1 peptides can precipitate PCNA from cell extracts.	*off-topic.*
Inhibition of SV40 DNA replication by p21PBP.	*off-topic.*
The minimum PCNA binding site on p21WAF1.	*not a fact.*
Critical residues within p21PBP.	*not a fact.*

Article #69

Authors: *Pyrowolakis G, Hartmann B, Muller B, Basler K, Affolter M.*
Title: *A simple molecular complex mediates widespread BMP-induced repression during Drosophila development.*
PMID: *15296719.*

Heading	ACE
Mad and Medea Directly Bind to a Dpp Morphogen-Dependent Silencer Element of the brk Gene.	*off-topic.*
The Spacing but Not the Sequence between the Mad and the Med Binding Site Is Important for Shn Recruitment.	*off-topic.*
Shn Is a Modular Repressor Protein.	*perfect:* Shn is a subtype of Modular-Repressor-Protein.
Functional Mad/Med/Shn-Dependent Silencers Are Found in Other Drosophila Genes.	*off-topic.*
Germline Stem Cells Are Maintained by Shn Recruitment to an SE in the bam Gene.	*off-topic.*
Dpp Directly Represses gsb Transcription in the Dorsal Ectoderm.	*off-topic.*

Article #70

Authors: *Maeda T, Gupta MP, Stewart AF.*
Title: *TEF-1 and MEF2 transcription factors interact to regulate muscle-specific promoters.*
PMID: *12061776.*

Heading	ACE
Physical interaction between TEF-1 and MEF2.	*perfect:* TEF-1 physically interacts-with MEF2.
Functional interaction between TEF-1 and MEF2.	*perfect:* TEF-1 functionally interacts-with MEF2.
TEF-1 isoforms differentially modulate MEF2 activation of the MLC2v and MHC promoters.	*no match*[2]*.*
MEF2 recruits a TEF-1-like factor to the MLC2v promoter.	*no match*[4]*.*

Article #71

Authors: *Authors*
Title: *Title*
PMID: *PMID*

Heading	ACE
Binding of PML to Multiple Corepressors and HDAC1.	*partial*[4]*:* PML binds HDAC1.
Colocalization of Corepressors and PML.	*no match*[4]*.*
PML Is Involved in Mad-Mediated Transcriptional Repression.	*off-topic.*
PML-RARa Inhibits the Mad-Mediated Repression.	*off-topic.*

Article #72

AUTHORS: *Hayashi H, Abdollah S, Qiu Y, Cai J, Xu YY, Grinnell BW, Richardson MA, Topper JN, Gimbrone MA Jr, Wrana JL, Falb D.*

TITLE: *The MAD-related protein Smad7 associates with the TGFbeta receptor and functions as an antagonist of TGFbeta signaling.*

PMID: *9215638.*

Heading	ACE
Smad7 Inhibits TGFbeta Signaling.	*partial*[1]*:* Smad7 inhibits the activity of TGF-Beta.
Smad7 Blocks Activation of Smad2.	*partial*[1]*:* Smad7 inhibits the activity of Smad2.
Smad7 Interacts Stably with the TGFbeta Receptor.	*perfect:* Smad7 stably interacts-with a receptor of TGF-Beta.
A Nonfunctional Mutant of Smad7 Does Not Interact with the TGFbeta Receptor.	*perfect:* A mutant of Smad7 does not interact-with a receptor of TGF-Beta.

Article #73

AUTHORS: *Nagy L, Kao HY, Chakravarti D, Lin RJ, Hassig CA, Ayer DE, Schreiber SL, Evans RM.*

TITLE: *Nuclear receptor repression mediated by a complex containing SMRT, mSin3A, and histone deacetylase.*

PMID: *9150137.*

Heading	ACE
SMRT Has Two Independent Repressor Domains.	*perfect:* There is a repressor-domain X of SMRT and there is a repressor-domain Y of SMRT and X is not Y.
Interaction between SMRT and mSin3A.	*perfect:* SMRT interacts-with MSin3A.
An SMRT–mSin3A–HDAC1 Ternary Complex.	*perfect:* There is a protein-complex X such that SMRT is a subunit of X and MSin3A is a subunit of X and HDAC1 is a subunit of X.
Functional Interaction between HDAC1 and SMRT.	*perfect:* HDAC1 functionally interacts-with SMRT.
Retinoic Acid and Trichostatin A Synergize in Cell Differentiation.	*off-topic.*

Article #74

AUTHORS: *Zong H, Li Z, Liu L, Hong Y, Yun X, Jiang J, Chi Y, Wang H, Shen X, Hu Y, Niu Z, Gu J.*

TITLE: *Cyclin-dependent kinase 11(p58) interacts with HBO1 and enhances its histone acetyltransferase activity.*

PMID: *15963510.*

Heading	ACE
CDK11p58 interacts with histone acetyltransferase HBO1 in yeast.	*perfect:* CDK11p58 interacts-with HBO1 in Yeast.
CDK11p58 binds to HBO1 in vitro.	*perfect:* CDK11p58 binds HBO1 in-vitro.
CDK11p58 associates with HBO1 in mammalian cells.	*perfect:* CDK11p58 associates-with HBO1 in Mammal.
CDK11p58 colocalizes with HBO1 in the nucleus.	*perfect:* CDK11p58 colocalizes-with HBO1 in Nucleus.
CDK11p58 enhances the histone acetyltransferase activity of HBO1 in vitro.	*partial*[1]*:* CDK11p58 increases the activity of HBO1 in-vitro.
CDK11p58 enhances the histone acetyltransferase activity of HBO1 in vivo.	*partial*[1]*:* CDK11p58 increases the activity of HBO1 in-vivo.

Article #75

AUTHORS: *Chi RJ, Olenych SG, Kim K, Keller TC 3rd.*

TITLE: *Smooth muscle alpha-actinin interaction with smitin.*

PMID: *15833278.*

Heading	ACE
Alpha-Actinin-smitin binding in vitro.	*perfect:* Alpha-Actinin binds Smitin in-vitro.
The native alpha-actinin rod domain interacts with smitin.	*partial*[1]*:* The rod-domain of Alpha-Actinin interacts-with Smitin.
Alpha-Actinin rod and C-terminal domains bind smitin.	*perfect:* The c-terminal domain of Alpha-Actinin binds Smitin and a rod-domain of Alpha-Actinin binds Smitin.
The alpha-actinin R2-R3 spectrin-like repeat and C-terminus domains out compete native alpha-actinin for smitin binding.	*no match*[4]*.*
Effects of PIP2 on interactions of alpha-actinins with cardiac muscle titin and smooth muscle smitin.	*not a fact.*

Article #76

Authors: *Bubeck Wardenburg J, Pappu R, Bu JY, Mayer B, Chernoff J, Straus D, Chan AC.*
Title: *Regulation of PAK activation and the T cell cytoskeleton by the linker protein SLP-76.*
PMID: *9846482.*

Heading	ACE
In Vitro Interaction of Nck and SLP-76.	*perfect:* Nck interacts-with SLP-76 in-vitro.
In Vivo Interaction of Nck and SLP-76 in Jurkat and Normal Peripheral T Cells.	*perfect:* Nck interacts-with SLP-76 in-vivo in Jurkat-T-Cell. Nck interacts-with SLP-76 in-vivo in Normal-Peripheral-T-Cell.
Formation of a Tri-Molecular Complex Consisting of SLP-76, Nck, and Vav.	*perfect:* There is a protein-complex X such that SLP-76 is a subunit of X and Nck is a subunit of X and Vav is a subunit of X.
Integration of Vav GEF Activity with Rho-GTPase Effector Protein Function by the Tri-Molecular Complex.	*no match*[4]*.*
Regulation of F-Actin Formation by SLP-76, Vav, and Nc.	*no match*[4]*.*

Article #77

Authors: *Ivanova AV, Ivanov SV, Zhang X, Ivanov VN, Timofeeva OA, Lerman MI.*
Title: *STRA13 interacts with STAT3 and modulates transcription of STAT3-dependent targets.*
PMID: *15223310.*

Heading	ACE
STRA13 interacts with STAT3beta, the short isoform of STAT3.	*perfect:* STRA13 interacts-with STAT3Beta.
STAT3beta associates with the HLH and the C-terminal regions of STRA13.	*partial*[1]*:* STAT3Beta associates-with the c-terminal region of STRA13.
STRA13 binds to the tyrosine-phosphorylated forms of STAT3beta and STAT3alpha proteins.	*partial*[1]*:* STRA13 binds STAT3Beta and binds STAT3Alpha.
STRA13 activates transcription from STAT-responsive elements GAS, ISRE and STAT3.	*off-topic.*
The basic domain and the C-terminal region of STRA13 are responsible for transcription activation from cytokine-responsive elements.	*off-topic.*
STRA13 expression is induced by cytokines.	*off-topic.*
Roles of STRA13 and STAT3 in the regulation of Fas transcription.	*not a fact.*
STRA13 over-expression causes apoptosis.	*off-topic.*

Article #78

Authors: *Li W, Kedersha N, Chen S, Gilks N, Lee G, Anderson P.*
Title: *FAST is a BCL-X(L)-associated mitochondrial protein.*
PMID: *15110758.*

Heading	ACE
Subcellular localization of endogenous FAST.	*not a fact.*
Identification of a mitochondrial targeting domain.	*not a fact.*
FAST interacts with BCL-XL.	*perfect:* FAST interacts-with BCL-XL.

Article #79

AUTHORS: *Muromoto R, Sugiyama K, Yamamoto T, Oritani K, Shimoda K, Matsuda T.*
TITLE: *Physical and functional interactions between Daxx and TSG101.*
PMID: *15033475.*

Heading	ACE
Association of Daxx with DMAP1 and TSG101 in 293T cells.	*perfect:* DAXX associates-with DMAP1 in 293T-Cell and associates-with TSG101 in 293T-Cell.
Co-localization of Daxx and TSG101 in the nucleus.	*perfect:* DAXX colocalizes-with TSG101 in Nucleus.
TSG101 and Daxx cooperatively repress glucocorticoid receptor-mediated transcriptional activity.	*off-topic.*

Article #80

AUTHORS: *Kim YY, Park BJ, Seo GJ, Lim JY, Lee SM, Kimm KC, Park C, Kim J, Park SI.*
TITLE: *Long form of cellular FLICE-inhibitory protein interacts with Daxx and prevents Fas-induced JNK activation.*
PMID: *14637155.*

Heading	ACE
Overexpression of c-FLIPL but not of c-FLIPS renders cells resistant to Fas-induced apoptosis.	*off-topic.*
c-FLIPL inhibits the Fas-induced JNK activation.	*partial*[1]*:* C-FLIPL inhibits the activity of JNK.
c-FLIPL but not c-FLIPS interacts with Daxx.	*perfect:* C-FLIPL interacts-with DAXX. C-FLIPS does not interact-with DAXX.
Daxx interacts with c-FLIPL through its Fas-binding domain.	*perfect:* A domain X of DAXX binds Fas and the domain X interacts-with C-FLIPL.

Article #81

AUTHORS: *Mikolajczyk M, Shi J, Vaillancourt RR, Sachs NA, Nelson M.*
TITLE: *The cyclin-dependent kinase 11(p46) isoform interacts with RanBPM.*
PMID: *14511641.*

Heading	ACE
Identification of RanBPM as CDK11p46-interacting protein.	*perfect:* RanBPM interacts-with CDK11p46.
Binding of RanBPM with CDK11p46 in vitro and in vivo.	*perfect:* RanBPM binds CDK11p46 in-vitro and binds CDK11p46 in-vivo.
CDK11p46 phosphorylates RanBPM in vitro	*perfect:* CDK11p46 phosphorylates RanBPM in-vitro.

Article #82

AUTHORS: *Fleckenstein DS, Dirks WG, Drexler HG, Quentmeier H.*
TITLE: *Tumor necrosis factor receptor-associated factor (TRAF) 4 is a new binding partner for the p70S6 serine/threonine kinase.*
PMID: *12801526.*

Heading	ACE
Identification of TRAF4 as a new binding partner of p70S6K.	*perfect:* TRAF4 binds P70S6K.
Activated lymphotoxin-beta receptor induces p70S6K/TRAF4 interaction.	*no match*[2]*.*
Crosstalk between the TNF-receptor and the PI3K/p70S6K signaling pathway.	*off-topic.*
TRAF4 inhibits Fas-inducible apoptosis in HEK-293 cells.	*off-topic.*

Article #83

AUTHORS: *Barberis M, Pagano MA, Gioia LD, Marin O, Vanoni M, Pinna LA, Alberghina L.*
TITLE: *CK2 regulates in vitro the activity of the yeast cyclin-dependent kinase inhibitor Sic1.*
PMID: *16168390.*

Heading	ACE
Sic1 protein binds to CK2alpha and CK2beta subunits.	*perfect:* Sic1 binds CK2Alpha that is a subunit of CK2 and binds CK2Beta that is a subunit of CK2.
Sic1 is phosphorylated by both the alpha subunit and the holoenzyme CK2 in vitro.	*perfect:* CK2Alpha phosphorylates Sic1 in-vitro. CK2 phosphorylates Sic1 in-vitro.
Phosphorylation of the CK2 consensus site within a Sic1-derived peptide increases binding to the mammalian Cdk2/cyclin A complex	*no match*[1].
Sic1 fully phosphorylated by CK2 is a stronger inhibitor of the S-Cdk activity than the unphosphorylated protein.	*no match*[1].
Homology modelling of the Sic1/Cdk1/Clb5 ternary complex.	*not a fact.*
The CK2 consensus site of Sic1 is predicted to interact with the Cdk1/Clb5 kinase complex.	*no match*[3].

Article #84

Authors: *Irie S, Hachiya T, Rabizadeh S, Maruyama W, Mukai J, Li Y, Reed JC, Bredesen DE, Sato TA.*
Title: *Functional interaction of Fas-associated phosphatase-1 (FAP-1) with p75(NTR) and their effect on NF-kappaB activation.*
PMID: *10544233.*

Heading	ACE
FAP-1 interacts with the C-terminal SPV of p75NTR in vitro.	*partial*[4]*:* FAP-1 interacts-with the c-terminus of P75NTR in-vitro.
FAP-1 interacts with p75NTR in vivo.	*perfect:* FAP-1 interacts-with P75NTR in-vivo.
Co-localization of FAP-1 and p75NTR.	*perfect:* FAP-1 colocalizes-with P75NTR.
Interaction of FAP-1 with p75NTR is involved in the regulation of TRAF6-mediated NF-kappaB activation.	*no match*[2].
p75NTR C-terminal mutation (V to M) enhances its pro-apoptotic activity	*no match*[1].

Article #85

Authors: *Roder K, Wolf SS, Larkin KJ, Schweizer M.*
Title: *Interaction between the two ubiquitously expressed transcription factors NF-Y and Sp1.*
PMID: *10393239.*

Heading	ACE
NF-YA and Sp1 interact in-vivo.	*perfect:* NFYA interacts-with Sp1 in-vivo.
An Sp1 interaction domain is located between amino acids 55 and 139 of NF-YA.	*no match*[1].
An NF-YA interaction domain is located between amino acids 139 and 344 of Sp1.	*no match*[1].
In-vitro interaction of Sp1 with CBF-B.	*perfect:* Sp1 interacts-with CBFB in-vitro.
NF-YA co-immunoprecipitates Sp1 from rat hepatoma cells.	*no match*[1].

Article #86

Authors: *Yang X, Khosravi-Far R, Chang HY, Baltimore D.*
Title: *Daxx, a novel Fas-binding protein that activates JNK and apoptosis.*
PMID: *9215629.*

Heading	ACE
Two-Hybrid Screen for Novel Fas-Interacting Proteins.	*not a fact.*
Cloning of Daxx cDNA and Northern Analysis.	*not a fact.*
Daxx Interacts with Fas Both In Vitro and In Vivo.	*perfect:* DAXX interacts-with Fas in-vitro and interacts-with Fas in-vivo.
Daxx Potentiates Fas-Mediated Apoptosis.	*no match*[1].
Daxx Activates the JNK/SAPK Pathway.	*off-topic.*
Deletion Mutagenesis of Daxx.	*not a fact.*
DaxxC Is a Dominant-Negative Inhibitor of Fas-Mediated Apoptosis and JNK Activation.	*no match*[1].
Daxx and FADD Define Two Distinct Fas-Mediated Signaling Pathways.	*off-topic.*

Article #87

AUTHORS: *Shu HB, Halpin DR, Goeddel DV.*
TITLE: *Casper is a FADD- and caspase-related inducer of apoptosis.*
PMID: *9208847.*

Heading	ACE
Identification of Casper.	*not a fact.*
Induction of Apoptosis by Casper and Its Protease-like Domain.	*off-topic.*
A Deletion Mutant of Casper Blocks TNF- and Fas-Induced Apoptosis.	*no match*[1].
Casper Interacts with Distinct Signaling Proteins.	*partial*[3]*:* Casper interacts-with Signaling-Protein.
Casper Interacts with FADD and Is Recruited to Fas.	*partial*[1]*:* Casper interacts-with FADD.
Casper Interacts with Caspase-8 through Distinct Domains.	*partial*[3]*:* Casper interacts-with Caspase-8.
CrmA Interacts with Caspase-8 but Not Casper or Caspase-3.	*perfect:* CrmA interacts-with Caspase-8 and does not interact-with Casper and does not interact-with Caspase-3.
Casper Indirectly Induces Caspase-3 Activity.	*partial*[1]*:* Casper regulates the activity of Caspase-3.
Casper Interacts with Caspase-3.	*perfect:* Casper interacts-with Caspase-3.
Casper Is Proteolytically Processed in Mammalian Cells.	*no match*[4].
Casper Interacts with TRAF1 and TRAF2.	*perfect:* Casper interacts-with TRAF1 and interacts-with TRAF2.

Article #88

AUTHORS: *Hsu H, Huang J, Shu HB, Baichwal V, Goeddel DV.*
TITLE: *TNF-dependent recruitment of the protein kinase RIP to the TNF receptor-1 signaling complex.*
PMID: *8612133.*

Heading	ACE
Identification of RIP as a TRADD-Interacting Protein.	*perfect:* RIP interacts-with TRADD.
RIP Interacts with TRAF Proteins.	*perfect:* RIP interacts-with TRAF.
Characterization of RIP Deletion Mutants.	*not a fact.*
The Death Domain of RIP Blocks TNF-Mediated NF-kappaB Activation.	*partial*[2]*:* The death-domain of RIP inhibits the activity of NF-kappaB.
TRADD Recruits RIP to TNFR1.	*no match*[1].
TNFR1–TRADD–TRAF2–RIP Complex.	*perfect:* There is a protein-complex X such that TNFR1 is a subunit of X and TRADD is a subunit of X and TRAF2 is a subunit of X and RIP is a subunit of X.
Association of RIP with TNFR1 Is TNF Dependent.	*partial*[2]*:* RIP associates-with TNFR1.
RIP Is a SerineThreonine Protein Kinase.	*perfect:* RIP is a subtype of Serine-Threonine-Protein-Kinase.

Article #89

AUTHORS: *Chatellier J, Hartley O, Griffiths AD, Fersht AR, Winter G, Riechmann L.*
TITLE: *Interdomain interactions within the gene 3 protein of filamentous phage.*
PMID: *10606756.*

Heading	ACE
Specific interaction between g3p-D12 and g3p-D3.	*perfect:* G3p-D12 specifically interacts-with G3p-D3.
Functional significance of the g3p-D12/g3p-D3 interaction.	*not a fact.*
Implications for selective infection of phage (SIP) strategies.	*not a fact.*
Implications for phage infectivity.	*not a fact.*

Appendix C

Ontology Lexicon Converter

This appendix contains the code for the Ontology Lexicon Converter that translates the Ontology Lexicon Format into the ACE Lexicon Format. The Ontology Lexicon Converter is written in SWI Prolog.

```
% ----------------------------------------------------------------------
% Ontology Lexicon Converter
% version 1.0, 3 January 2006
% Tobias Kuhn
% ----------------------------------------------------------------------
% This program converts from the Ontology Lexicon Format into the ACE
% Lexicon Format.
% ----------------------------------------------------------------------

:- module(ontology_lexicon_converter,
          [
             transform/2  % +InputFile, +OutputFile
          ]).


% ----------------------------------------------------------------------
% transform(+InputFile, +OutputFile)
% ----------------------------------------------------------------------
% Transforms a file in the Ontology Lexicon Format (InputFile) into a
% file in the ACE Lexicon Format (OutputFile).
% ----------------------------------------------------------------------

transform(InputFile, OutputFile) :-
   open(InputFile, read, In),
   open(OutputFile, write, Out),
   repeat,
      read(In, OLFTerm),
      process_term(Out, OLFTerm),
   !,
   close(In),
   close(Out),
   check_references.


% ----------------------------------------------------------------------
% entry(-OLFTerm)
% ----------------------------------------------------------------------
% This dynamic predicate is used to store the processed OLF terms.
```

```
% ----------------------------------------------------------------------

:- dynamic(entry/1).


% ----------------------------------------------------------------------
% reference(-Ref, -Type)
% ----------------------------------------------------------------------
% This dynamic predicate is used to store the references and their
% types.
% ----------------------------------------------------------------------

:- dynamic(reference/2).


% ----------------------------------------------------------------------
% process_term(+OutputStream, +OLFTerm)
% ----------------------------------------------------------------------
% Transforms the term OLFTerm into the ACE Lexicon Format and writes it
% in the stream OutputStream. After that it fails, unless OLFTerm is
% end_of_file.
% ----------------------------------------------------------------------

% End Of File
process_term(_Out, end_of_file) :-
   !.


% CONCEPTS

% Common Noun
process_term(Out, OLFTerm) :-
   OLFTerm = concept(id(ID),
                     cn(singular(Singular),
                        type(ObjectType),
                        gender(Gender)),
                     superconcepts(SuperconceptList)),
   valid_id(ID),
   atom(Singular),
   valid_object_type(ObjectType),
   valid_gender(Gender),
   list_of_atoms(SuperconceptList),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperconceptList, concept),
   writeq(Out, cn(singular(Singular),
                  plural(''),
                  singular_aliases([]),
                  plural_aliases([]),
                  type(ObjectType),
                  gender(Gender),
                  group(countable),
                  comment(''))),
   write(Out, '.'), nl(Out),
   fail.

% Common Noun (with reference)
process_term(_Out, OLFTerm) :-
   OLFTerm = concept(id(ID),
                     cn(ref(RefID)),
                     superconcepts(SuperconceptList)),
   atom(RefID),
```

```
   entry(role(id(RefID),
              cn(singular(_Singular),
                 type(_ObjectType),
                 gender(_Gender)),
              _Superroles,
              _Domain,
              _Range)),
   valid_id(ID),
   list_of_atoms(SuperconceptList),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperconceptList, concept),
   fail.

% Adjective
process_term(Out, OLFTerm) :-
   OLFTerm = concept(id(ID),
                     adj(positive(Positive)),
                     superconcepts(SuperconceptList)),
   valid_id(ID),
   atom(positive),
   list_of_atoms(SuperconceptList),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperconceptList, concept),
   writeq(Out, adj(positive(Positive),
                   comparative(''),
                   superlative(''),
                   positive_aliases([]),
                   comparative_aliases([]),
                   superlative_aliases([]),
                   complementing_preposition(''),
                   comment(''))),
   write(Out, '.'), nl(Out),
   fail.

% Intransitive Verb
process_term(Out, OLFTerm) :-
   OLFTerm = concept(id(ID),
                     iv(third_singular(ThirdSingular),
                        third_plural(ThirdPlural),
                        phrasal_particle(PhrasalParticle)),
                     superconcepts(SuperconceptList)),
   valid_id(ID),
   atom(ThirdSingular),
   atom(ThirdPlural),
   atom(PhrasalParticle),
   list_of_atoms(SuperconceptList),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperconceptList, concept),
   writeq(Out, iv(third_singular(ThirdSingular),
                  third_plural(ThirdPlural),
                  third_singular_aliases([]),
                  third_plural_aliases([]),
                  type(unspecified),
                  phrasal_particle(PhrasalParticle),
                  comment(''))),
   write(Out, '.'), nl(Out),
   fail.
```

```
% INDIVIDUALS

% Propername
process_term(Out, OLFTerm) :-
   OLFTerm = individual(id(ID),
                        pn(singular(Singular),
                           type(ObjectType),
                           gender(Gender))),
   valid_id(ID),
   atom(Singular),
   valid_object_type(ObjectType),
   valid_gender(Gender),
   !,
   assert(entry(OLFTerm)),
   writeq(Out, pn(singular(Singular),
                  plural(''),
                  singular_aliases([]),
                  plural_aliases([]),
                  type(ObjectType),
                  gender(Gender),
                  comment(''))),
   write(Out, '.'), nl(Out),
   fail.


% ROLES

% Transitive Verb (basic version)
process_term(Out, OLFTerm) :-
   OLFTerm = role(id(ID),
                  tv(third_singular(ThirdSingular),
                     third_plural(ThirdPlural),
                     phrasal_particle(PhrasalParticle),
                     direct_preposition(DirectPreposition)),
                  superroles(SuperroleList),
                  domain(Domain),
                  range(Range)),
   valid_id(ID),
   atom(ThirdSingular),
   atom(ThirdPlural),
   atom(PhrasalParticle),
   atom(DirectPreposition),
   list_of_atoms(SuperroleList),
   atom(Domain),
   atom(Range),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperroleList, role),
   assert(reference(Domain, concept)),
   assert(reference(Range, concept)),
   writeq(Out, tv(third_singular(ThirdSingular),
                  third_plural(ThirdPlural),
                  third_singular_aliases([]),
                  third_plural_aliases([]),
                  type(unspecified),
                  phrasal_particle(PhrasalParticle),
                  direct_preposition(DirectPreposition),
                  comment(''))),
   write(Out, '.'), nl(Out),
   fail.

% Transitive Verb (extended version)
```

```
process_term(Out, OLFTerm) :-
   OLFTerm = role(id(ID),
                  tv(third_singular(ThirdSingular),
                     third_plural(ThirdPlural),
                     phrasal_particle(PhrasalParticle),
                     direct_preposition(DirectPreposition)),
                  superroles(SuperroleList),
                  domain(Domain),
                  range(Range),
                  context(ContextList)),
   valid_id(ID),
   atom(ThirdSingular),
   atom(ThirdPlural),
   atom(PhrasalParticle),
   atom(DirectPreposition),
   list_of_atoms(SuperroleList),
   atom(Domain),
   atom(Range),
   valid_context_list(ContextList),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperroleList, role),
   assert(reference(Domain, concept)),
   assert(reference(Range, concept)),
   assert_context_references(ContextList),
   writeq(Out, tv(third_singular(ThirdSingular),
                  third_plural(ThirdPlural),
                  third_singular_aliases([]),
                  third_plural_aliases([]),
                  type(unspecified),
                  phrasal_particle(PhrasalParticle),
                  direct_preposition(DirectPreposition),
                  comment(''))),
   write(Out, '.'), nl(Out),
   fail.

% Adverb
process_term(Out, OLFTerm) :-
   OLFTerm = role(id(ID),
                  adv(adverb(Adverb),
                      type(ModifierType)),
                  superroles(SuperroleList),
                  domain(Domain),
                  range(Range)),
   valid_id(ID),
   atom(Adverb),
   valid_modifier_type(ModifierType),
   list_of_atoms(SuperroleList),
   atom(Domain),
   atom(Range),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperroleList, role),
   assert(reference(Domain, concept)),
   assert(reference(Range, concept)),
   writeq(Out, adv(adverb(Adverb),
                   adverb_comparative(''),
                   adverb_superlative(''),
                   adverb_aliases([]),
                   adverb_comparative_aliases([]),
                   adverb_superlative_aliases([]),
                   type(ModifierType),
```

```
                  comment(’’))),
   write(Out, ’.’), nl(Out),
   fail.

% Of-Construct
process_term(Out, OLFTerm) :-
   OLFTerm = role(id(ID),
                  cn(singular(Singular),
                     type(ObjectType),
                     gender(Gender)),
                  superroles(SuperroleList),
                  domain(Domain),
                  range(Range)),
   valid_id(ID),
   atom(Singular),
   valid_object_type(ObjectType),
   valid_gender(Gender),
   list_of_atoms(SuperroleList),
   atom(Domain),
   atom(Range),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperroleList, role),
   assert(reference(Domain, concept)),
   assert(reference(Range, concept)),
   writeq(Out, cn(singular(Singular),
                  plural(’’),
                  singular_aliases([]),
                  plural_aliases([]),
                  type(ObjectType),
                  gender(Gender),
                  group(countable),
                  comment(’’))),
   write(Out, ’.’), nl(Out),
   fail.

% Of-Construct (with reference)
process_term(_Out, OLFTerm) :-
   OLFTerm = role(id(ID),
                  cn(ref(RefID)),
                  superroles(SuperroleList),
                  domain(Domain),
                  range(Range)),
   atom(RefID),
   entry(concept(id(RefID),
                 cn(singular(_Singular),
                    type(_ObjectType),
                    gender(_Gender)),
                 _Superconcepts)),
   valid_id(ID),
   list_of_atoms(SuperroleList),
   atom(Domain),
   atom(Range),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperroleList, role),
   assert(reference(Domain, concept)),
   assert(reference(Range, concept)),
   fail.

% Comparative Form of Adjective
process_term(Out, OLFTerm) :-
```

```
   OLFTerm = role(id(ID),
                  adj(comparative(Comparative)),
                  superroles(SuperroleList),
                  domain(Domain),
                  range(Range)),
   valid_id(ID),
   atom(Comparative),
   list_of_atoms(SuperroleList),
   atom(Domain),
   atom(Range),
   !,
   assert(entry(OLFTerm)),
   assert_references(SuperroleList, role),
   assert(reference(Domain, concept)),
   assert(reference(Range, concept)),
   writeq(Out, adj(positive(''),
                comparative(Comparative),
                superlative(''),
                positive_aliases([]),
                comparative_aliases([]),
                superlative_aliases([]),
                complementing_preposition(''),
                comment(''))),
   write(Out, '.'), nl(Out),
   fail.



% Invalid Term
process_term(_Out, OLFTerm) :-
   write('ERROR. Invalid Entry: '),
   writeq(OLFTerm), nl,
   fail.


% ----------------------------------------------------------------------
% assert_references(+ReferencesList, +Type)
% ----------------------------------------------------------------------
% Stores the references in ReferencesList which is a list of atoms. Type
% is one of {context, role} and defines the type of the reference.
% ----------------------------------------------------------------------

assert_references([], _Type).

assert_references([Ref|Rest], Type) :-
   assert(reference(Ref, Type)),
   assert_references(Rest, Type).


% ----------------------------------------------------------------------
% assert_context_references(+ContextRefList)
% ----------------------------------------------------------------------
% ContextRefList is a list of context-terms that look like prep(Prep,
% Concept). This predicate stores the concept references.
% ----------------------------------------------------------------------

assert_context_references([]).

assert_context_references([prep(_Prep, ConceptID)|Rest]) :-
   assert(reference(ConceptID, concept)),
   assert_context_references(Rest).
```

```
% ----------------------------------------------------------------------
% check_references
% ----------------------------------------------------------------------
% Checks whether the stored references are valid. This predicate
% succeeds always. For each unresolved reference a warning-message is
% printed onto the standard output-device.
% ----------------------------------------------------------------------

check_references :-
   reference(Ref, Type),
   \+ id_exists(Ref, Type),
   write('WARNING. Undefined Reference: '),
   writeq(Ref), nl,
   fail.

check_references.


% ----------------------------------------------------------------------
% valid_id(+ID)
% ----------------------------------------------------------------------
% Checks whether ID is a valid ID, i.e. is atomic and unique.
% ----------------------------------------------------------------------

valid_id(ID) :-
   atom(ID),
   \+ id_exists(ID, _Type).


% ----------------------------------------------------------------------
% id_exists(+ID, ?Type)
% ----------------------------------------------------------------------
% Checks whether the ID exists for the given type. Type is one of
% {individual, concept, role}.
% ----------------------------------------------------------------------

id_exists(ID, individual) :-
   entry(individual(id(ID), _Word)).

id_exists(ID, concept) :-
   entry(concept(id(ID), _Word, _Superconcepts)).

id_exists(ID, role) :-
   entry(role(id(ID), _Word, _Superroles, _Domain, _Range)).

id_exists(ID, role) :-
   entry(role(id(ID), _Word, _Superroles, _Domain, _Range, _Context)).


% ----------------------------------------------------------------------
% valid_object_type(?ObjectType)
% ----------------------------------------------------------------------
% Checks whether ObjectType is a valid object-type in the ACE Lexicon
% Format.
% ----------------------------------------------------------------------

valid_object_type(unspecified).
valid_object_type(person).
valid_object_type(object).
valid_object_type(time).
```

```
% ----------------------------------------------------------------------
% valid_gender(?Gender)
% ----------------------------------------------------------------------
% Checks whether Gender is a valid gender description in the ACE Lexicon
% Format.
% ----------------------------------------------------------------------

valid_gender(human).
valid_gender(neutr).
valid_gender(masc).
valid_gender(fem).


% ----------------------------------------------------------------------
% valid_modifier_type(?ModifierType)
% ----------------------------------------------------------------------
% Checks whether ModifierType is a valid modifier-type in the ACE
% Lexicon Format.
% ----------------------------------------------------------------------

valid_modifier_type(unspecified).
valid_modifier_type(manner).
valid_modifier_type(time).
valid_modifier_type(location).
valid_modifier_type(duration).
valid_modifier_type(frequency).
valid_modifier_type(instrument).
valid_modifier_type(destination).
valid_modifier_type(comitative).


% ----------------------------------------------------------------------
% list_of_atoms(+List)
% ----------------------------------------------------------------------
% Succeeds if List is a list of atomic values.
% ----------------------------------------------------------------------

list_of_atoms([]).

list_of_atoms([Atom|Rest]) :-
   atom(Atom),
   list_of_atoms(Rest).


% ----------------------------------------------------------------------
% valid_context_list(+List)
% ----------------------------------------------------------------------
% Succeeds if List is a valid context list in the ACE Ontology Format.
% ----------------------------------------------------------------------

valid_context_list([]).

valid_context_list([prep(Prep, Concept)|Rest]) :-
   atom(Prep),
   atom(Concept),
   valid_context_list(Rest).
```

Bibliography

[1] *ACE Lexicon Specification*. University of Zurich, Department of Informatics. 22 October 2005.
`http://www.ifi.unizh.ch/attempto/tools/documentation/ace_lexicon.html`

[2] *APE Webservice*. University of Zurich, Department of Informatics. 26 October 2005.
`http://www.ifi.unizh.ch/attempto/tools/documentation/ape_webservice.html`

[3] Franz Baader, Werner Nutt. *Basic Description Logics*. In "The Description Logic Handbook: Theory, Implementation, and Applications". Cambridge University Press. 2003.

[4] Franz Baader, Ian Horrocks, Ulrike Sattler. *Description Logics as Ontology Languages for the Semantic Web*. Theoretical Computer Science, RWTH Aachen; Department of Computer Science, University of Manchester. 2003.

[5] Alex Borgida. *On the Relative Expressiveness of Description Logics and Predicate Logics*. Department of Computer Science. Rutgers University. New Brunswick. 1996.

[6] Francesco M. Donini. *Complexity of Reasoning*. In "The Description Logic Handbook: Theory, Implementation, and Applications". Cambridge University Press. 2003.

[7] Norbert E. Fuchs, Uta Schwertel, Rolf Schwitter. *Attempto Controlled English – Not Just Another Logic Specification Language*. University of Zurich, Department of Informatics. 1998.

[8] Norbert E. Fuchs, Stefan Hoefler, Kaarel Kaljurand, Gerold Schneider, Uta Schwertel. *Discourse Representation Structures of ACE 4 Sentences*. University of Zurich, Department of Informatics. 2005.

[9] The Gene Ontology Consortium. *Gene Ontology: tool for the unification of biology*. In "Nature America Inc.". 2000.

[10] Thomas R. Gruber. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. Stanford Knowledge Systems Laboratory, Palo Alto. 1993.

[11] Hans Kamp, Uwe Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht / Boston / London. 1993.

[12] Robert Kowalski. *Logic without Model Theory*. In "What is a Logical System?". Oxford University Press. 1994.

[13] Deborah L. McGuinness, Frank van Harmelen. *OWL Web Ontology Language Overview*. W3C World Wide Web Consortium. 10 February 2004. `http://www.w3.org/TR/2004/REC-owl-features-20040210/`

[14] Daniele Nardi, Ronald J. Brachman. *An Introduction to Description Logics*. In "The Description Logic Handbook: Theory, Implementation, and Applications". Cambridge University Press. 2003.

[15] Rolf Schwitter. *Controlled Natural Language as Interface Language to the Semantic Web*. Centre for Language Technology, Maquarie University, Sydney. 2005.

[16] Rolf Schwitter, Anna Ljungberg, David Hood. *ECOLE: A Look-ahead Editor for a Controlled Language*. Centre for Language Technology, Maquarie University, Sydney. 2003.

[17] Barry Smith, Werner Ceusters, Bert Klagges, Jacob Köhler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan L. Rector, Cornelius Rosse. *Relations in biomedical ontologies*. Genome Biology 2005, Volume 6, Issue 5.

[18] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co, Pacific Grove, CA. 1999.

[19] Mike Uschold, Michael Gruninger. *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review, Volume 11, Number 2. 1996.

[20] Jan Wielemaker. *SWI-Prolog 5.4, Reference Manual*. University of Amsterdam, Department of Social Science Informatics. 2004.