A Formal Approach to Aggregated Belief Formation

Annerieke Heuvelink[1,2], Michel C.A. Klein[1], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
[2] TNO Defence, Security, and Safety, Department of Training and Instruction,
P.O. Box 23, 3769 ZG Soesterberg, The Netherlands
{a.heuvelink,michel.klein,treur}@few.vu.nl

Abstract. This paper introduces a formal method to aggregate over basic beliefs, in order to deduce aggregated or complex beliefs as often used in applications. Complex beliefs can represent several things, such as a belief about a period in which other beliefs held or the minimal or maximal certainty with which a belief held. As such they contain richer information than the basic beliefs they are aggregated from and can be used to optimize an agent's search through its memory and its reasoning processes. The developed method can also aggregate over aggregated beliefs, hence nested aggregations are possible. An implementation in Prolog demonstrates its operationality.

Keywords: Belief Aggregation, Term Algebra, Memory.

1 Introduction

Agents in applications commonly store beliefs about the state of the world in what is often called a world model or belief base. This belief base is usually a set of atomic beliefs that grows over time. There are several potential problematic issues related to such a belief base. First, after some time the size of the belief base can result in the practical problem that retrieval and inferences will become time expensive: the time needed will grow with the size of the belief base. Second, some inferences result in intermediate results that might be useful again at a later point in time. When the intermediate results are not stored, they have to be recalculated, while when they are stored, they will add to the size problem. Third, this way of storing beliefs seems not very similar to the human way of using memory. For example, humans often forget specific details, but still remember aggregated abstractions or consequences of specific facts. Taking this as a point of departure, it may be explored how aggregated beliefs can be formed and stored within an agent as new entities.

Fact is that aggregations can be formed from many different perspectives and at multiple levels. Which perspectives are chosen and which level of aggregation is needed, is application and task depended. Therefore a general approach that distinguishes all possible types of aggregations one by one, may become quite complex; for example, if m different aggregation types are possible, and n levels of aggregation, then the number of aggregation types is m^n, which already for relatively low numbers such as m = 10 and n = 5 leads to a high number (100.000) of aggregated beliefs. To

avoid this explosion, in this paper an algebraic approach is adopted that distinguishes a general notion of aggregation operator that (1) is parameterised by the specific constraint that is used in an aggregation process, and (2) can be used in a recursive manner. Thus the combinatorics induced by different levels is replaced by term expressions that can be formed by nesting a number of (parameterised) variants of the aggregation operator.

The presented formalism allows for the specification of complex beliefs at a higher level of aggregation than the basic atomic beliefs. Such aggregated belief representations have the advantage that they are often closer to the level of aggregation that is used in specification of reasoning steps, and are therefore often more useful. For example, it is more convenient to specify reasoning steps based on an aggregated belief such as the "last most certain belief", than based on a long list of atomic beliefs at different time points and with different certainties. Moreover, some aggregations are used several times. In this case, the aggregation functions as a reasoning template that specifies how a new belief can be deduced from other beliefs. This template only has to be specified once and can be reused later on.

The remainder of the paper is structured as follows. First, section 2 introduces an application domain and the basic belief formalism that is used. In the subsequent section, the algebraic approach to belief aggregation is described, which is formalised as a term algebra in section 4. Section 5 and 6 demonstrate the operationality of the approach, by presenting a Prolog implementation and showing how the algebra allows the formation of several useful complex beliefs, which are defined as specific aggregations. Section 7 relates the work to other research, both in the area of knowledge compilation and temporal abstraction. Finally, in section 8 the research is summarised and future research plans exposed.

2   Belief Formalism

In [1] a software agent was developed that compiles a tactical picture of its environment, which entails the classification and identification of surface radar contacts. For modelling its behaviour the need was identified to explicitly represent the time at which a belief was held by the agent in its short term memory (STM). In other words: when it believed it. The main reason to represent this, was to enable the (biased) reasoning over (possible inconsistent) beliefs over time [1]. For example, when at time t it is believed that the position of a contact is [x1, y1], while at time t + n it is believed to be position [x2, y2], the average speed of the contact can be inferred. This is useful because the speed on a contact might contain information concerning its identity, e.g., large ships that are neutral usually do not sail faster than 20 knots. In the same way a contact's manoeuvring pattern can be inferred, which is relevant as it gives away much information concerning a contact's intent.

In order to logically represent other aspects, namely uncertainty of information, and the fact that information can come from various sources, every belief also received a source and a certainty label. As a result, the basic knowledge entity of the agent is represented by *belief(P, O, V, T, S, C)*, which denotes the belief that the indeterminable property P holds for object O with the value V at time T, based on source S and with certainty C. An example belief denoting that it is believed at time 8 with

high certainty that the identity of the radar contact1 is friendly because of radio con-servation is: *belief (identity, contact1, friendly, 8, radio, 0.9)*.

The value, time, and certainty label of beliefs about a specific property and object are often used to reason about trends in those beliefs, which can lead to new beliefs. For example, a new belief can be formed about a contact being a merchant, and there-fore neutral, due to beliefs about it sailing in a straight line. The certainty of the belief that the contact is a merchant is determined by the period, as by the certainty, with which it is believed that it does this. For the deduction of other beliefs it is often im-portant to deduce what the last, or most (un)certain belief about a specific something was. For example, the highest certainty with which it was once believed that a contact fired is relevant for deducing whether it might be hostile.

The beliefs formed by the agent over time are stored in the agent's belief base, rep-resenting long term memory (LTM). When storing beliefs in LTM, it is important to denote when they were formed or retrieved in STM. For this a new reference to time is introduced, the two-place predicate *holds_at*. When the basic belief predicate of the object language is reified to a term *b*, the time at which the belief is held in STM can be expressed by *holds_at(b, t)*. For every *belief(p, o, v, t, s, c)* that is found in the agent's belief base it holds that *holds_at(belief(p,o,v,t,s,c), t),* since the t of the belief denotes the time it was formed (was present in STM).

3   Belief Aggregation

Unfortunately, the storage of time-stamped beliefs led to the problematic issues men-tion in section 1 [2]. Therefore, this paper focuses on the development of a generic, formal approach to the formation of arbitrary aggregations over these basic beliefs, to form all kinds of so-called aggregated or complex beliefs. Complex beliefs abstract or cluster information of the lower level. They form a solution for keeping the amount of time required to search through the agent's belief base within limits. Furthermore, they can be used to model specific properties of human memory, like the forgetting of specific details.

3.1   Aggregation Examples

An example of a complex belief that an agent can form was mentioned in section 2, namely a belief about the period during which a certain belief held. That complex belief, about the duration of the straight manoeuvre of a contact, can be used directly to infer a new belief, e.g., that that contact might be a merchant.

While specifying the formal model underlying the reasoning and behaviour of the cognitive agent described in [1], it was found that often specific types of information are required to deduce new beliefs. To be precise, often the last, earliest, most certain or uncertain, increasing in certainty, or longest held belief was required. In addition, it was noticed that the deduction processes of several of these beliefs are very similar, e.g. the deduction of a last belief (belief with highest T) is very similar to the deduc-tion of a most-certain belief (belief with highest C).

These observations spurred the development of an generic approach to belief aggre-gation in which a complex belief is defined as an aggregation that takes the form of a constraint (e.g., highest) that must hold for a certain variable (e.g., T) of a certain more

of less specified belief (e.g., belief(identity, contact1, friendly, T, S, C) in which the P, O, and V are specified while the T, S, and C are left variable). The term algebra formalizing this approach to the formation of aggregated beliefs is introduced in the section 4, while the section after that discusses an implementation of the approach in Prolog.

3.2 Complex Belief of Type Integrated Sources

The *integrated_sources* belief was the most important complex belief the developed agent in [1] reasoned with instead of with its basic beliefs. This complex belief represents which value is currently believed by the agent to hold for a certain property and object, and with which certainty. To determine this, inconsistencies formed by beliefs from different sources, with different certainties, and held at different times, have to be resolved. Much research has been done on how to deal with such inconsistencies, see e.g. [3, 4].

In [1] a relative simple procedure was introduced to determine which value V was currently believed to held with certainty C by an agent for a given P and O. This procedure takes into account that a belief's validity over time is strongly influenced by its predicate type (property) P. Values of some predicates are much more persistent than others; consider the chance that a contact's position, speed, or intent changes over time. The following logical expression denotes the meaning of the complex belief called *integrated_sources*:

$$given\ (p,\ o)$$
$$\forall v1\ \forall t1\ \forall s1\ \forall c1\ \forall t\ \forall pd \qquad [$$
$$\quad holds\_at(\ complex\_belief($$
$$\qquad integrated\_sources,\ for(p,\ o),\ has\_values(v1,\ c1-pd*(t-t1))\ ),\ t)$$
$$\leftrightarrow$$
$$\quad holds\_at(\ complex\_belief(\ last,\ for(p,\ o,\ s1),\ has\_values(v1,\ t1,\ c1)\ ),\ t)\ \wedge$$
$$\quad persistence\_decay(p,\ pd)\ \wedge$$
$$\quad \neg\exists s2\ \exists v2\ \exists t2\ \exists c2 \qquad [$$
$$\qquad holds\_at(\ complex\_belief(\ last,\ for(p,\ o,\ s2),\ has\_values(v2,\ t2,\ c2)\ ),\ t)\ \wedge$$
$$\qquad c2-pd*(t-t2) > c1-pd*(t-t1)\ ] \qquad] \qquad (1)$$

This expression specifies that the agent believes at time t that for a given P and O, *for(p, o)*, the value *v1* holds, which is the value of the belief about P and O whose certainty is the greatest after taking into account the time passed since it was formed and the persistence of the property; $c1 - pd * (t - t1)$. This might entail that the value of an older belief with a certain certainty is believed over the value of a newer belief that has a lower certainty. It might also be the other way around; it depends on the nature (persistence) of the property. In this expression another complex belief was used of the type *last*, which has as exact definition:

$$\forall p\ \forall o\ \forall v\ \forall t\ \forall s\ \forall c\ \forall n \qquad [$$
$$\quad holds\_at(\ complex\_belief(\ last,\ for(p,\ o,\ s),\ has\_values(\ v,\ t,\ c)\ ),\ n)$$
$$\leftrightarrow$$
$$\quad holds\_at(\ belief(p,\ o,\ v,\ t,\ s,\ c),\ t)\ \wedge t\leq n\ \wedge$$
$$\quad \neg\exists t'\ \exists v'\ \exists c' \qquad [$$
$$\qquad holds\_at(\ belief(p,\ o,\ v',\ t',\ s,\ c'),\ t')\ \wedge t'\geq t\ \wedge t'\leq n\ ]\ ] \qquad (2)$$

Expression 2 specifies that the agent believes at time n that *t* is the last time at which a belief incorporating the given P, O, and S, *for(p, o, s)*, held. This is the case since *t* is the time label of a belief with that given P, O, and S, for which it holds that no other belief exists with the same P, O, and S, but a higher T (*t'*). This complex belief of type *last* is defined as an aggregation of all the beliefs with the given P, O, and S, and the constraint Highest for their time label T. Besides a specification for T, this aggregation also specifies the free variables V and C. This is a quite standard aggregation, considering the limited complexity of the constraint that it takes into account. The aggregation as which the complex belief of type *integrated_sources* is defined, is much less standard. The constraint that has to be taken into account in that aggregation is much more specific and not likely to be reusable, see section 5.

The current paper focuses on the development of an algebraic approach to have an efficient representation of aggregated beliefs. For demonstration purposes it elaborates on several possible types of these aggregated beliefs, which are defined as specific aggregations. Notice that the introduced aggregations simply serve as examples, and that many more are possible. The approach is set up in such a generic way that all kinds of constraints that lead to all kinds of complex beliefs can be expressed with it.

4   Algebraic Formalization

The algebra specification of the aggregation functions on beliefs is defined by a basic ontology, by means of which its objects and relations can be expressed. The primitive terms used in the algebra are defined by a many-sorted signature. The signature takes into account symbols for *sorts*, *constants*, *functions* and *relations*. Examples of *sorts* are: LABEL, CONSTRAINT, TIME, TYPE, AGGREGATIONBASE, AGGREGATEDBELIEF, ARGUMENTLIST, BASICBELIEFBASE, PROPERTY, or OBJECT. *Constants* are names of objects within sorts; examples are 'speed', '20', or 'fast'. *Functions* denote mappings from a (combination of) sort(s) to another sort; examples of function symbols are agg, holdsat, + and *. *Relations* symbols (relating different sorts) used are, for example = and <. Logical relationships involve conditional statements involving relations. Figure 1 depicts a large part or the algebra specification with the definitions used listed below. Arrows with no label are defined by 'e' which denotes (injective) embedding.

agg: LABEL   x   CONSTRAINT   x   AGGREGATEDBELIEF   →
    AGGREGATIONNAME
e: AGGREGATIONBASE   → AGGREGATEDBELIEF
e: BASICBELIEFBASE   → AGGREGATEDBELIEF
e: COMPLEXBELIEFBASE   → AGGREGATEDBELIEF
holdsat: AGGREGATIONNAME x TIME → AGGREGATIONBASE
definedas: COMPLEXBELIEFBASE x AGGREGATIONBASE
holdsat: COMPLEXBELIEF x TIME → COMPLEXBELIEFBASE
complexbelief: TYPE   x   ARGUMENTLIST   x   RANGELIST   →
    COMPLEXBELIEF
e: PROPERTY   → ARGUMENTLIST
e: OBJECT   → ARGUMENTLIST
e: VALUE   → ARGUMENTLIST
e: TIME   → ARGUMENTLIST
e: SOURCE   → ARGUMENTLIST

e: CERTAINTY   → ARGUMENTLIST
e: RANGE   → RANGELIST
holdsat: BASICBELIEF x TIME → BASICBELIEFBASE
belief: PROPERTY x OBJECT x VALUE x TIME x SOURCE x CERTAINTY →
    BASICBELIEF
abstraction1: LABELTYPE x VAR → LABEL
abstraction2: LABELTYPE x LABELTYPE x VAR x VAR → LABEL
constraint: NAME x VARIABLE x AGGREGATEDBELIEF → CONSTRAINT
forall, exists: VAR x FORMULA → FORMULA
definedas: CONSTRAINT x FORMULA
not: FORMULA → FORMULA
and, or, implies: FORMULA x FORMULA → FORMULA
e: ATOM → FORMULA
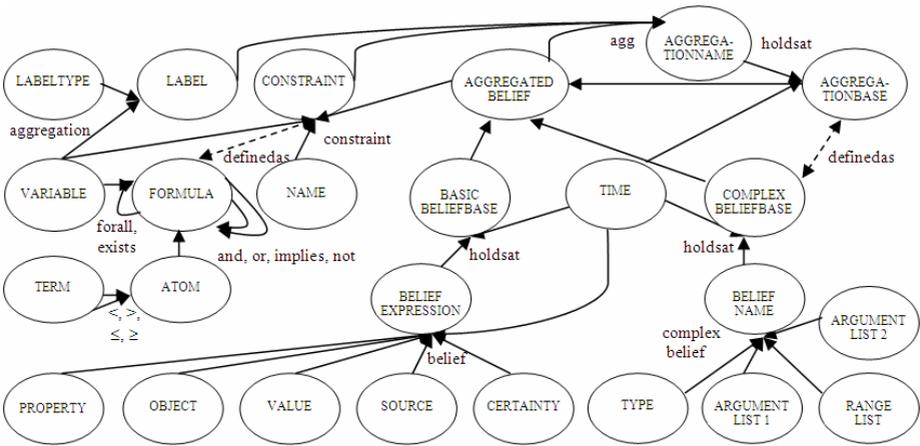$<, >, \leq, \geq$: TERM x TERM → ATOM

Fig. 1. Overview of the algebra for belief aggregation

A number of sorts are considered primitive; they only contain constants such as names and values: LABELTYPE, VARIABLE, PROPERTY, OBJECT, VALUE, TIME, SOURCE, CERTAINTY, RANGE, and TYPE. Some other sorts are more or less standard, and/or may depend on application dependent functions: ATOM, TERM, FORMULA.

Sort ARGUMENTLIST 1 contains terms listing 6 arguments with at each of the 6 positions instances. Two special instances exist; *free* and *range*, which denote that the argument of that position is variable. The sort RANGELIST contains terms listing the 6 ranges for the 6 arguments of ARGUMENTLIST 1. The range is only relevant for the arguments with the special instance *range*. In the case of a normal instance the corresponding range is *nr* (not relevant) while in the case of the special instance *free*, the corresponding term is *any*. Sort ARGUMENTLIST 2 contains terms listing 6 arguments with at each of the 6 positions instances. Two special instances exist; *given* and *nr*, which denote respectively that the argument of that position was already specified in ARGUMENTLIST 1, or is no longer relevant given the TYPE.

Note that the function agg can be used in a recursive manner together with the function holdsat. The nested term structures that result, represent beliefs at different levels of aggregation: the level is the number of nested agg functions occurring in the term.

The area of algebraic specification has a long history. From the extensive literature techniques can be borrowed to obtain an implementation of calculations in the algebra, for example in a functional or logic programming language. If relations are involved, an implementation has to take into account both functional and logical aspects; e.g., [5, 6]. Following this tradition, the next section introduces an implementation of the developed algebra in the logic programming language Prolog.

The algebra is considered a term algebra, which specifies the different variations of aggregated belief expressions that can be formed. The current Prolog implementation generates such expressions, but does not perform evaluations of whether two different expressions should be considered as having the same content or meaning. In future work it is planned to extend this approach to an algebra for which also equations are specified, and an implementation where such equations are incorporated.

5   Implementation

The algebra of section 4 is implemented in SWI-Prolog [7]. In this section, the implementation choices are explained. For the readability of this section, only parts of the Prolog program are shown. The complete source code can be downloaded from: http://www.few.vu.nl/~heuvel/CIA-AggregationAlgebra.pl

5.1   Controlling Aggregations

The current implementation does not incorporate automatic control of aggregations. Instead two 'programs' are implemented that can be called from the Prolog-shell: *holds* and *post*. The *holds* program is shown below and can be used to request the results of a specific aggregation, or to request the values for which a specific complex belief holds. Notice that complex beliefs are defined as aggregations and are as such interchangeable. When no holds_at attribute is included in the query, it is assumed that the query requests the result of the aggregation or complex belief at the current time. When a holds_at is included, the query requests the result of the aggregation or complex belief that holds at the specified time.

```holds(B):- B = complex_belief(_, _, _, _), current_time(N), complex_belief_is_defined_as(holds_at(B,N), holds_at(agg(L,C,A),N)), holds_at(agg(L,C,A),N).```	Query about B, B is a complex belief, and is checked for the current time N. The definition of the complex belief B is the aggregation agg(L, C, A), which is requested for time N.
```holds(B):- B = holds_at(X,N), X = complex_belief(_, _, _, _), complex_belief_is_defined_as(B, holds_at(agg(L,C,A),N)),```	Query about B, B is whether X holds at time N, with X being a complex belief. The complex belief X within B is defined as an aggregation.

| ```
    is_time(N),
    holds_at(agg(L,C,A),N).
``` | When N is an actual time, that aggregation is requested for N. |
| ```
holds(B):-
    B = agg(L,C,A),
    current_time(N),
    holds_at(agg(L,C,A),N).
``` | Query about B, B is an aggregation, and is checked for the current time N, and therefore requested at N. |
| ```
holds(B):-
    B = holds_at(agg(L,C,A),N),
    is_time(N),
    holds_at(agg(L,C,A),N).
``` | Query about B, B is whether agg(L,C,A) holds at N, when this is an actual time agg(L,C,A) is requested at N. |

The *holds* program does not alter the belief-base and as such can be used to investigate 'what-if' questions. Besides the *holds* program also a *post* program exists whose procedure is almost identical, except each of its rules is extended with the extra condition *assert(_)*. This adds the complex belief, defined as the aggregation that is checked to hold at N, to the belief base.

5.2 Free and Bounded Variables

Complex beliefs do not have 6 atomic arguments like basic beliefs, but are made up of four arguments. The first of these is atomic and specifies a name for the complex belief, which is also referred to as its *type*. The latter three arguments are predicates; each embeds 6 arguments whose positions respectively represent the P, O, V, T, S, and C. So a complex belief is represented by *complex_belief(Type, For(…), With_Ranges(…), Has_Values(…))*, which denotes that it is believed by the agent that for that complex_belief *Type* and for the given constants in *For*, taken into account the *With_Ranges* in which the free variables in *For* have to lie, the constants in *Has_Values* count.

An example complex belief denoting that it is believed that the last time at which a belief was held about the hostile identity of contact1 is 7, and that was with a certainty 0.6 and based on radio contact is: *complex_belief (last, for(identity, contact1, hostile, free, free, free), with_ranges(nr, nr, nr, any, any, any), counts(given, given, given, 7, radio, 0.6))*. Such a complex belief is the result of the agent reasoning about what the last time, i.e. highest T, was that it believed that the identity of contact1 was hostile. When it would have reasoned about what the last time was that it believed with less than 0.5 certainty that that was the case, the following complex belief might have hold: *complex_belief (last, for(identity, contact1, hostile, free, free, range), with_ranges(nr, nr, nr, any, any, [0, 0.5]), has_values(given, given, given, 4, vision, 0.4))*.

Given this representation of complex beliefs, an example of a complex_belief_ is_defined_as relation which defines a complex belief as a specific aggregation is:

```
complex_belief_is_defined_as(
  holds_at( complex_belief(
      last,
      for(P,O,V,free,S,C),
      with_ranges(nr,nr,nr,any,nr,nr),
      has_values(given,given,given,X,given,given)),N),
```

```
holds_at( agg(temporal_aggregation(T),
              highest_free(X,any,P,O,V,S,C),
              holds_at(belief(P,O,V,T,S,C),N)),N)).
```

The aggregation shown here will return the highest T that it can find for the given P, O, V, S, and C. When it does not matter what the S and C are, but it is required to find the highest (last) T that is now restrained to a certain time range [Tb, Te] for a given P, O, and V, the following aggregation is applicable:

```
complex_belief_is_defined_as(
   holds_at(   complex_belief(
      last,
      for(P,O,V,range,free,free),
      with_ranges(nr,nr,nr,[Tb,Te],any, any),
      has_values(given,given,given,X,Y,Z)),N),
   holds_at(   agg(
      temporal_source_certainty_aggregation(T,S,C),
      highest_range_free_free
         (X,[Tb,Te],Y,any,Z,any,P,O,V),
      holds_at(belief(P,O,V,T,S,C),N)),N)).
```

This aggregation will return the highest T that it can find for the given P, O, and V. The S and C that it returns are those of the belief with that highest T. This aggregation example demonstrates that variables can be free or that they can be restricted to a specific range. When it is checked whether a certain aggregation holds at a certain time the following clause executes:

```
holds_at(agg(L,C,A),_):-
     term_variables(L,V),
     constraint_is_defined_as
        (constraint(C,A,V),F),
     F.
```

To determine the agg(L, C, A) at time _, the variables in L are listed in V. The definition of the constraint C for the subject A and the variables in V is F, which is consequently requested.

The first condition term_variables(L,V) is a built-in Prolog predicate that unifies V with a list of variables, each corresponding with a unique variable of L and ordered in order of appearance in L. So for the example above it holds:

?- term_variables(temporal_certainty_source_aggregation(T,C,S),V).
V=[G34,G35,G27], T=G34, C=G35, S=G27.

The second condition is a user-defined predicate that defines what the constraint C entails for the aggregated belief A with its free variables listed in V; namely F, which forms the last condition. On the next page, an example constraint_is_defined_as is shown for the constraint that is required to deduce the first complex belief of type *last* introduced in this section. Notice that this *highest_free* constraint can be reused, e.g., to deduce a complex belief of type *surest* when it is combined with a certainty_aggregation. Its logical expression is:

given A,
$\forall x \quad [highest\_free (x, any) \leftrightarrow A(x) \land \forall x1 [A(x1) \rightarrow x1 \leq x]]$ (3)

```
constraint_is_defined_as(          Definition  (
constraint(                        constraint (
  highest_free(X,any,F1,F2,F3,F4,F5),   C,
  A1, [X1]),                            A, V   ),
and( copy_term( (A1, X1, F1, F2, F3, F4, F5),
                (A, X, F1, F2, F3, F4, F5)),  F        ).
     and(A, forall(A1, X1 =< X)))).
```

The constraint that was required to deduce the second complex belief of type *last* introduced in this section, is shown next. It can be seen that this constraint only considers options A1 whose values X1 for the variable X lies within the range [Xb, Xe] specified for it.

```
constraint_is_defined_as(              Definition  (
constraint(                            constraint (
  highest_range_free_free(
      X,[Xb,Xe],Y,any,Z,any,F1,F2,F3),    C,
  A1, [X1, Y1, Z1]),                       A, V   ),
and( copy_term( (A1,X1,Y1,Z1,F1,F2,F3),
                (A,X,Y,Z,F1,F2,F3)),       F
  and(A, and( X>=Xb, and( X<Xe,
    forall( and(A1, and( X1>=Xb, X1<Xe))            ).
      X1 =< X)))))).
```

5.3 Nested Aggregations

The reason that in the constraint_is_defined_as Prolog clauses the values F1, ..., Fn are embedded is that although they are usually instantiated, they do not have to be. When they are not, and are left out of the query, they do get instantiated when Prolog requests A. However, when next is asked whether for all X1 in A1 X1 ≤ X holds, this probably fails. This is because the left-out variable that now is instantiated in A, is still free in A1, so much more A1's are checked than there should be.

The reason why variables are allowed to exist in places where atoms are expected is because this freedom enables nested aggregations. An example of a nested aggregation is the complex belief *integrated_sources* introduced in section 3:

```
complex_belief_is_defined_as(
  holds_at(complex_belief(
              integrated_sources,
              for(P,O,free,free,free,free),
              with_ranges(nr,nr,any,any,any,any),
              has_values(given,given,X,nr,nr,Y)),N),
  holds_at(agg(
              certainty_temporal_source_value_
                aggregation(C,T,S,V),
              highest_free_after_free_for_free_
                free_for_predicate_and_time
                Y,any,_,any,_,any,X,any,P,O,N),
              holds_at(complex_belief(
                last,
```

```
                for(P,O,free,free,S,free),
                with_ranges(nr,nr,any,any,nr,any),
                has_values(given,given,V,T,given,C)),N))

        ,N) )
```

In this clause a complex belief of type *last* functions as aggregated belief for the aggregation that deduces the complex belief of type *integrated_sources*. This latter aggregation aggregates over values, times, sources, and certainties of beliefs about a given property and object, in order to retrieve a specific value and certainty. The aggregation belief it needs as input is a complex belief of type *last* that aggregates over values, times and certainties for a given property, object and source. However, the latter (S) is not given but variable, because the top-aggregation needs this *last* type for all possible sources. Note that instead of the complex belief of type *last* also the aggregation as which it is defined could have been used as input.

The constraint used within the aggregation to deduce the complex belief *integrated_sources* is much more specific and therefore less reusable than, e.g., the *highest_free* constraint. These two examples nicely illustrate the reach of the proposed aggregation mechanism. In principle all possible constraints can be added and used to form new types of complex beliefs that in turn can be used in other aggregations.

6   Example Scenarios

From http://www.few.vu.nl/~heuvel/CIA-AggregationAlgebra.pl the source code of our Prolog program can be downloaded. In the case presented, an agent attempts to infer the identity of a radar contact. Information about this contact can be gathered by the radar as by the agent's own vision. Furthermore, the agent can generate new beliefs by reasoning over other beliefs. Over time the following basic beliefs have held in STM and are now stored in LTM:

```
holds_at(belief(identity, contact1, neutral, 2,
                vision, 0.5), 2).
holds_at(belief(identity, contact1, neutral, 3,
                radar, 0.3), 3).
holds_at(belief(speed, contact1, 20, 3,
                radar, 0.9), 3).
holds_at(belief(identity, contact1, hostile, 4,
                vision, 0.9), 4).
holds_at(belief(identity, contact1, hostile, 4,
                id_from_speed, 0.4), 4).
holds_at(belief(speed, contact1, 28, 6,
                radar, 0.9), 6).
holds_at(belief(speed, contact1, 30, 7,
                vision, 0.5), 7).
holds_at(belief(identity, contact1, hostile, 8,
                vision, 0.7), 8).
holds_at(belief(identity, contact1, hostile, 8,
                id_from_speed, 0.8), 8).
```

At current_time 10, two of the nine beliefs stored in the agent's LTM are formed by the agent's reasoning rule id_from_speed, which forms the source of those beliefs. At this moment the agent might start another reasoning process for which it requires the last belief about a hostile identity of contact1. This query results in:

```
1 ?- holds(complex_belief(last, for(identity,contact1,hostile,
free,free,free), with_ranges(nr,nr,nr,any,any,any), has_values
(given,given,given,T,S,C))).
T = 8,
S = vision,
C = 0.7 ;
T = 8,
S = id_from_speed,
C = 0.8 ;
fail.
```

By chance, two sets of atoms are found that both adhere to this query. In such case the agent might be interested in the surest one of these two last beliefs. This complex belief of type *surest_last* is formed by aggregating the label certainty_temporal_source_aggregation and the highest_free_free_free constraint with that complex belief of type *last* as aggregated belief. The query for complex belief of type *last_surest* yields a totally different result: it is an aggregation of the same constraint but in combination with a temporal_certainty_source_aggregation and on complex beliefs of the type *surest*.

```
2 ?- holds(complex_belief(surest_last, for(identity,contact1
,hostile,free,free,free), with_ranges(nr,nr,nr,any,any,any),
 has_values(given,given,given,T,S,C))).
T = 8,
S = id_from_speed,
C = 0.8 ;
fail.

3 ?- holds(complex_belief(last_surest, for(identity,contact1
,hostile,free,free,free), with_ranges(nr,nr,nr,any,any,any),
 has_values(given,given,given,T,S,C))).
T = 4,
S = vision,
C = 0.9 ;
fail.
```

Another possibility would be that the agent's superior asks the agent what it believes that contact1's identity is. At that moment the agent will retrieve its last beliefs about the identity of that contact and form an answer. In the current case the agent believes contact1 might be neutral based on what it saw of the vessel, as on the radar-emission-pattern it received from the contact. However, it also believes it might be hostile, due to its high speed. In order to give its superior an answer the agent has to form a belief about the contact's identity by integrating the retrieved last information about its identity from the different sources. Given that the persistence-decay of a contact's identity (see section 3) is 0, the agent reports it believes the contact to be hostile since it was most sure of that.

```
4 ?- holds(complex_belief(integrated_sources, for(identity
,contact1,free,free,free,free), with_ranges(nr,nr,any,any,
any,any), has_values(given,given,V,nr,nr,C))).
V = hostile,
C = 0.8 ;
fail.
```

The agent's superior could also have asked what the agent believes that the speed of contact1 is. Again the agent needs to integrate information from different sources and times. However, because the persistence-decay of speed is larger than 0, say 0.05, it also has to take into account how long ago it was that it believed that information. The answer it will give is 28, see below. This knowledge is deduced from the basic belief at time 6 that its speed was 28, but notice that the certainty with which it is believed has decayed; from 0.9 to 0.7. Moreover, a newer belief concerning the contact's speed existed. However, even though the predicate's certainty decreases over time, still the value of the older belief is believed because the certainty of the new belief was very low.

```
5 ?- holds(complex_belief(integrated_sources, for(speed,
contact1,free,free,free,free), with_ranges(nr,nr,any,any
,any,any), has_values(given,given,V,nr,nr,C))).
V = 28,
C = 0.7 ;
fail.
```

7   Related Research

The technique for pre-processing a knowledge base to derive intermediate conclusions that is presented in this paper is related to the area of knowledge compilation. Knowledge compilation is defined in [8] as "methods of processing off-line a knowledge base in such a way that the output of such a pre-processing can be used to speed up on-line answering for a class of queries, where the pre-processing should take an finite amount of time". Within the area of knowledge compilation a distinction is made between exact methods (which are sound and complete) and approximate methods, which either reduce the complexity by expressing the knowledge or query in a simpler language or by leaving out some (complex) parts of the knowledge base.

Our approach is an exact technique, as it only results in sound intermediate results. However, a difference with common techniques for knowledge compilation is that our method does not strive to derive all intermediate results, whereas knowledge compilation techniques usually try to find a representation of all theorems of the initial knowledge base. For example, they transform a knowledge base to normal form and compute all implicants or implicates. In contrast, our approach is driven by specific queries whose results are likely to be useful for the task execution. In that sense, our method is not complete, as it does not aim to represent all knowledge in a different representation. Moreover, our aggregations are usually more complex (and thus richer in information), whereas knowledge compilation techniques often result in simpler representations. Last, our aim is to compile new knowledge on-line instead of off-line.

Shahar [9] presents a framework for knowledge based temporal abstraction from time-stamped data. His formal specification of a domain's temporal-abstraction knowledge supports acquisition, maintenance, reuse, and sharing of that knowledge. His aim is partly the same as our, however, his framework allows for temporal abstractions only, whereas our algebra allows for arbitrary abstractions.

The area of belief revision is also related to our work. In belief revision, the question is how existing beliefs are influenced when new pieces of information are taken into account, for example when information is added, removed or changed. The dominant theory on belief revision, the so-called AGM model [10], formulates properties that an operator that performs revision should satisfy in order for being considered rational. Similarly, related work on belief merging focuses on the consequences of combining belief bases for the integrity of a belief base, for example, see [11]. In our work the logical consequences of the aggregations are not relevant, as no new knowledge is added. There is no inconsistent information that is merged and it does not happen that old information changes because all information is time-stamped. This is comparable to what Sripada [12] describes, who also uses time-stamped beliefs. He proposes a technique for the efficient revision of beliefs in knowledge bases for real-time applications, but only looked at binary beliefs.

Another type of related work is formed by approaches for memory storage in existing (cognitive) agent architectures. In a recent review study on computer-based human behaviour representations [13] it was generalized that "all the (human behaviour) models can represent either short term memory (STM) or long term memory (LTM)." However, the ways in which these memories function differ greatly. For example, ACT-R's STM is formed by a retrieval buffer that can hold one chunk, which it retrieves using an activation function from its declarative memory module (LTM) [14], while Soar's STM is formed by its working memory that is not limited in the number of elements it can hold [15]. Related to the differences in memories, differences exist in the representation of the declarative information entities stored in such modules. These representations range from nodes in a network with an activation value to first-order propositions.

The functioning of the various memories are in general fixed and tuned to bring about the behaviour for which the architecture was developed. No existing architecture is build to specifically deal with time-labelled constructs, let alone in the algebraic approach as introduced in this paper. Despite this, it might be possible to map the specific belief construct to the memory construct of an architecture, prohibited the form of the latter has a certain degree of freedom [2, 16]. Moreover, the constraints that are needed to infer required (possibly domain-specific) aggregations have to be implemented in the architecture as well, as the aggregation algebra.

8   Summary and Future Research

In this paper a method and a term algebra is presented to form arbitrary aggregations of beliefs in a knowledge base. The aggregations can be formed at different levels and from different perspectives, i.e. time aggregations, source aggregations, certainty aggregations, etc. A Prolog program is used to illustrate the feasibility of the approach. The motivation of this work is twofold: it should help to improve the computational problems when reasoning over a knowledge base, and it should reflect a more human way of storing information in memory. As such, the goal is to 'validly'

represent aggregations of humans over beliefs, both conscious as subconscious, which can be used in agent applications where agents should behave in a human-like way.

Up to now, the control of the formation of aggregations is not yet implemented. Future research will investigate the control of aggregations from two perspectives. The first will be inspired by the human processes of forgetting and remembering, the second by the human processes of attention and focusing in task execution.

References

1. Heuvelink, A., Both, F.B.: A Cognitive Tactical Picture Compilation Agent. In: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007), pp. 175–181. IEEE Computer Society Press, Los Alamitos (2007)
2. Both, F., Heuvelink, A.: From a Formal Cognitive Task Model to an Implemented ACT-R Model. In: Proceedings of the 8th International Conference on Cognitive Modeling (ICCM 2007), pp. 199–204. Psychology Press (2007)
3. Castelfranchi, C.: Representation and Integration of Multiple Knowledge Sources: Issue and Questions. In: Cantoni, V., Di Gesù, V., Setti, A., Tegolo, D. (eds.) Human & Machine Perception: Information Fusion, pp. 235–254. Plenum Press (1997)
4. Bloch, I., Hunter, A., et al.: Fusion: General Concepts and Characteristics. International Journal of Intelligent Systems 16(10), 1107–1134 (2001)
5. Drosten, K.: Translating Algebraic Specifications to Prolog Programs: a Comparative Study. In: Algebraic and Logic Programming. LNCS, vol. 343, pp. 137–146. Springer, Heidelberg (1988)
6. Hanus, M.: The Integration of Functions into Logic Programming: From Theory to Practice. Journal of Logic Programming 19, 20, 583–628 (1994)
7. Wielemaker, J.: An Overview of the {SWI-Prolog} Programming Environment. In: Proceedings of the 13th International Workshop on Logic Programming Environments, pp. 1–16 (2003)
8. Cadoli, M., Donini, F.M.: A Survey on Knowledge Compilation. AI Communications 10(3-4), 137–150 (1997)
9. Shahar, Y.: A Framework for Knowledge-based Temporal Abstraction. Artificial Intelligence 90(11), 79–133 (1997)
10. Konieczny, S., Pino Pérez, R.: Merging Information Under Constraints: A Logical Framework. Journal of Logic and Computation 12(5), 773–808 (2002)
11. Alchourròn, C.E., Gärdenfors, P., Makinson, D.: On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. Journal of Symbolic Logic 50, 510–530 (1985)
12. Sripada, S.M.: A Temporal Approach to Belief Revision in Knowledge Bases. In: Proceedings of the Ninth Conference on Artificial Intelligence for Applications, pp. 56–62. IEEE Computer Society Press, Los Alamitos (1993)
13. Morrison, J.E.: A Review of Computer-Based Human Behavior Representations and Their Relation to Military Simulations. Institute for Defense Analyses, Paper P-3845 (2003)
14. Anderson, J.R., Lebiere, C.: The Atomic Components of Thought. Lawrence Erlbaum Associates, Mahwah (1998)
15. Laird, J.E., Newell, A., Rosenbloom, P.S.: SOAR: An Architecture for General Intelligence. Artificial Intelligence 33(1), 1–64 (1987)
16. Muller, T.J., Heuvelink, A., Both, F.: Comparison of Implementations of a Cognitive Model in Soar and ACT-R. In: Proceedings of the 6th International Workshop on From Agent Theory to Agent Implementation (AT2AI-6) (2008)