2009 International Conference on Computational Science and Engineering

Improving Interobserver Reliability by Artificial Intelligence Techniques in Behavioural Research

Arjen van Alphen[1], Tibor Bosse[2], Catholijn M. Jonker[3], and Francien Koeman[1]

[1] *Dog Therapeutic Research Institute De Roedel*
Chemin de la Bosette 7, 6690 Salmchateau, Belgium
info@deroedel.com

[2]*Vrije Universiteit Amsterdam, Department of Artificial Intelligence*
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
tbosse@few.vu.nl

[3] *Man-Machine Interaction Department, Fac. EEMCS, Delft University of Technology*
Mekelweg 4, 2628 CD Delft, The Netherlands
c.m.jonker@tudelft.nl

Abstract

Interobserver reliability and reproducibility are well known problems in experimental research within the social and behavioural sciences. We propose the use of formal techniques and tools to reduce this problem. To this end we extend standard research methods by transcribing video material in terms of basic score units, using automated tools to define in logic the more complex score units in terms of the basic score units, and to automatically check these complex score units against the transcripts. Furthermore, we use pilot experiments to determine the basic score units. We show that the proposed extension significantly improves interobserver reliability and reproducibility. An important additional benefit of our method is that the repository of annotations remains useful even if the researcher decides to test other complex score units that can be formulated in terms of the basic score units used to annotate the collected data.

1. Introduction

Within the social and behavioural sciences, reliability is a necessary but insufficient condition for validity [2]. This article focuses on interobserver reliability, which measures agreement between two or more subjects rating the same object, phenomenon, or concept, see e.g., [4, 7, 11]. Both random factors (e.g., the view is blocked, errors in coding) and systematic factors can cause lower interobserver reliability estimates. There might be many systematic factors that contribute to a low reliability, consider for example that the observers follow the protocol in different ways, or that the observers lack agreement on the criteria for a response. The last category can, for example, result in one observer consistently coding more of the behavior than a second observer. However, different observers might also have different understanding of the scoring unit itself. In our opinion, the standard methods for analyzing interobserver reliability do not pay enough attention to this last category. When analyzing this reliability the choice of observers is not challenged. That contrasts with remarks as made in [4], p.103: "But the principal concern of many researchers is the reliability of their basic data-acquisition system-a human observer-recorder." The literature then typically proceeds by showing how to determine the degree of interobserver reliability for two observers, not more.

Another problem that is not raised in the text book discussions on observer reliability is the reproducibility of research in relation to reliability. However, some case studies exist. For example, Malek et al., [5] determined 5-observer reliability based on the proportion of agreement and the values of the kappa coefficient. Myllyluoma and Duck [8] determined

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.44

inter-observer correspondence using correlation and intra-class correlation (ICC) analyses. In this article we propose an adaptation of the standard methods in the following sense.

- **Testing interobserver reliability in the pilot.** Before starting the annotations, do an experiment to check the variability with respect to the behaviours you would like people to score. If variability is low, the behaviour is "easy" to score. We call such behaviours basic. If variability is high, this is a complex behaviour. High variability inhibits reproducibility.
- **Splitting up complex behaviours into basic score units.** Complex behaviours have to be formulated in terms of basic behaviours. Interobserver reliability has to be determined for new basic behaviours.
- **Formalizing behaviours.** When interobserver reliability is high for all basic score units, and all complex behaviours have been formulated in terms of basic score units, both complex and basic behaviours have to represented in a formal language.
- **Experimentation and scoring.** After all the above preparations have been completed, the real experiment can be held. The observers only score basic behaviours.
- **Computer Tools Check Complex Behaviours.** After scoring the basic behaviours, the computer takes the formal definitions of the complex behaviours and checks those against the annotated data.

We claim that as a result the variability with respect to complex behaviours is low, as that depends only on the variability in basic behaviours. The computer does not add variability for the computation of the complex behaviours. Furthermore, if the researchers would want to check other complex behaviours, then as long as those behaviours are defined in terms of the basic score units, the checking can be done automatically, nobody has to return to the tapes. A case study[1] in dog behaviour shows the effectiveness of our method.

2. A Case Study: Dog Behaviour

In our case study (see [1]), dogs were presented with a search task in a testing arena of 5 by 10 meters, see Figure 1. Their task was to locate a goody (a smelly type of sausage dogs like) that was hidden underneath one of three cones. An experimenter presented the goody to the dog before hiding it. A blindfold kept the dogs from seeing where the goody is placed. The wind always carried the smell of the goody towards the baseline. The dog departs from the flag.

The hypothesis was that paw preference correlates to task performance. Paw preference refers to which front paw the dog uses first when departing to locate the goody. Good performance was defined informally by dog experts as "going in a near straight line to the goody pylon or taking the shortest route to the experimenter who presented the goody to the dog before hiding it under the goody pylon."

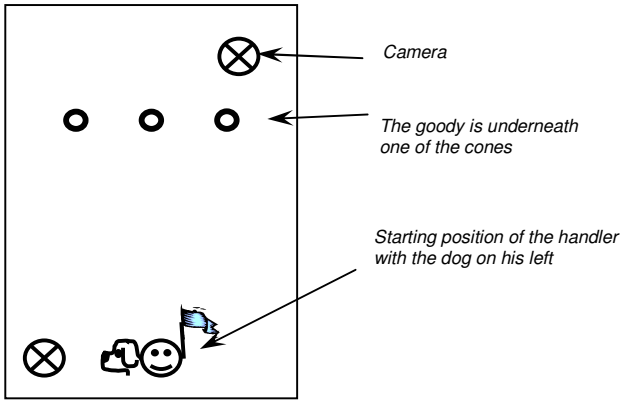

Figure 1. Setup of the experiment

3. Inter-Observer Reliability in Pilot Tests

From a first pilot experiment it proved difficult for human observers to establish whether or not the dog gave a good performance. To test the complexity of the "good performance" score unit that was intended to result in categorical data (bad vs good performance), we first conducted an experiment with 30 participants. We gave the participants the following definition of "good performance": the dog travels in a nearly straight line towards the middle dot. We then presented Figure 2 and asked in each group, who would consider A to show good performance, then the same for B, and then the same for C. The situations were chosen in such a way that A was clearly not good, C was clearly good, and, therefore, the real test concerned situation B, which is of much more interest than situation C as the last occurs rarely in practice. This resulted in an accumulated total of A=0, B=13, and C=30.

To establish interobserver reliability we adapted and then used the 2-observer proportion of agreement measure, see [4]. The proportion of agreement approach measures reliability by dividing the smaller of the two scores obtained for a session by the larger, and multiplying this ratio by 100, see e.g., [4]. For our multi-observer problem, we used the following formula's:

$$pa(i) = (N - min\_c(i)(score(c(i))))/N\% \quad (1)$$

$$ior = sum\_i(pa(i))/n \quad (2)$$

[1] Note that the presented case study is just an illustration: the approach is sufficiently generic to be applied in other domains in social science.

where *pa(i)* stands for proportion of agreement of trial *i*, *c* ranges over the possible categories for trial *i*, *score(c(i))* is the number of participants that used score trial *i* as belonging to category *c*, *ior* stands for the measure of interobserver reliability, and *n* stands for the number of trials. For the pilot this resulted in *pa(A)=100%*, *pa(B)=57%*, and *pa(C)=100%*. It made no sense to compute *ior* for these three trials, as trial B stood for the largest category of expected dog routes. In that sense trial B can be seen as an experiment by itself, with null-hypothesis that human observers would score B according to a normal distribution. The usual Chi-squared test for such situations clearly indicates that a score of 13 out of 30 is not enough to reject the null-hypothesis (χ^2=0.533, p<0.465). Hence, human judgment is not reliable with respect to score B, in fact human observers score B almost randomly.

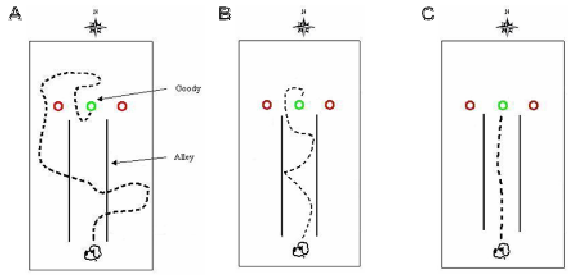

Figure 2. Pilot experiment

On the basis of these results, it was decided to find a more basic behaviour to score, upon which "good performance" could be defined. Although the property "good performance" is actually rather more complex, for the purpose of this paper, it suffices to say that we decided to test whether humans could reliably annotate the dog's route with directions according to the clock handles. After that for each of the situations in Figure 2 the same individuals of the pilot were asked to annotate the route the dogs with hours on the clock corresponding to the labeling of the example in Figure 3. Then we let the computer calculate "bad" or "good" performance per observer per trial. The results of this pilot experiment 2 were as follows: A=0, B=1, C=30. For which we calculated pa(A)=100% , pa(B)= 97%, and pa(C)= 100%. Here, obviously, also the interpersonal difference in judgment for score B (29 vs. 1) is significant (χ^2=26.133, p<0.001). Hence, that by letting the subjects annotate routes instead of the more complex behaviour of "good performance", the interobserver reliability went up from 57% to 97%.

The next part of the article concerns the use of tools and techniques to formally define basic and complex behaviours, The research assistants were given the formal definitions and asked to perform and annotate the real experiment. Of course interobserver reliability (for the 6 observers) was checked again.

The rest of this section is organized as follows. After the presentation of the formal language and definitions, the process of annotation and using an automated tool to check the complex behaviours against basic score units is explained.

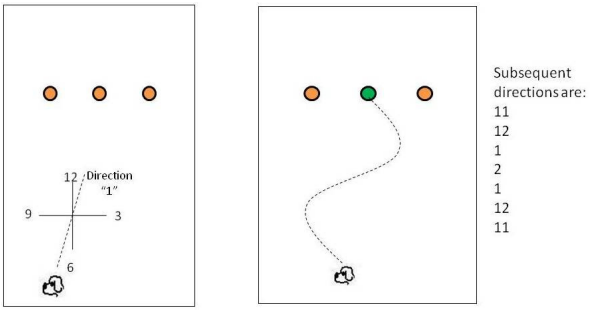

Figure 3. The example for directions

3.1 Identifying/Formalising Basic Behaviours

To formalise basic behaviours, first a formal ontology has to be created, by eliciting and formally representing domain knowledge. For the dog case study (in the context of the experiments of [1]), this was done after various discussions with domain experts, and using XML. In order to work with XML annotated files, a DTD (Document Type Definition) file must be generated in order the markup syntax of the XML file. The DTD file for the case study serves to illustrate important concepts for annotating behaviour:

```
<?xml version="1.0" encoding="UTF-8" ?>

<!ELEMENT transcript (data, general_circumstances, exercises)>

<!ELEMENT data (dog_number, name, breed, gender, age, castration,
      training*)>
<!ELEMENT dog_number (#CDATA)>
<!ELEMENT name (#CDATA)>
<!ELEMENT breed (#CDATA)>
<!ELEMENT gender (#CDATA)>
<!ELEMENT age (#CDATA)>
  <!ATTLIST age years CDATA #REQUIRED>
  <!ATTLIST age months CDATA #REQUIRED>
<!ELEMENT castration (#CDATA)>
<!ELEMENT training (#CDATA)>
  <!ATTLIST training arg1 CDATA #IMPLIED>

<!ELEMENT general_circumstances (date, area, surface, weather,
      disturbing_object*)>
<!ELEMENT date (#CDATA)>
  <!ATTLIST date year CDATA #REQUIRED>
  <!ATTLIST date month CDATA #REQUIRED>
  <!ATTLIST date day CDATA #REQUIRED>
<!ELEMENT area (#CDATA)>
<!ELEMENT surface (#CDATA)>
```

```
<!ELEMENT weather (temperature, wind_force, wind_direction, sun,
    precipitation)>
<!ELEMENT temperature (#CDATA)>
<!ELEMENT wind_force (#CDATA)>
<!ELEMENT wind_direction (#CDATA)>
<!ELEMENT sun (#CDATA)>
<!ELEMENT precipitation (#CDATA)>
  <!ATTLIST precipitation arg1 CDATA #IMPLIED>
<!ELEMENT disturbing_object (#CDATA)>
<!ELEMENT disturbing_noise (#CDATA)>
<!ELEMENT disturbing_smell (#CDATA)>
<!ELEMENT other_disturbance (#CDATA)>

<!ELEMENT exercises (exercise*)>
<!ELEMENT exercise (times*, comment*, exercise_content?)>
  <!ATTLIST exercise type (a | b | c | d | e) #REQUIRED>
  <!ATTLIST exercise trial (1 | 2 | 3 | 4) #REQUIRED>
  <!ATTLIST exercise index (i | ii | iii | iv | v | vi | rr1 | rr2 | rl1 | rl2 | ll1 | ll2 |
    lr1 | lr2) #REQUIRED>

<!ELEMENT times (starts, ends, on)>
<!ELEMENT starts (#CDATA)>
  <!ATTLIST starts min CDATA #REQUIRED>
  <!ATTLIST starts sec CDATA #REQUIRED>
<!ELEMENT ends (#CDATA)>
  <!ATTLIST ends min CDATA #REQUIRED>
  <!ATTLIST ends sec CDATA #REQUIRED>
<!ELEMENT on (#CDATA)>
  <!ATTLIST on cam CDATA #REQUIRED>

<!ELEMENT comment (#CDATA)>

<!ELEMENT exercise_content (atomic | sequential | parallel | comment)>
<!ELEMENT sequential (atomic | sequential | parallel | comment)+>
<!ELEMENT parallel (atomic | sequential | parallel | comment)+>
<!ELEMENT atomic (#CDATA)>
  <!ATTLIST atomic arg1 CDATA #IMPLIED>
  <!ATTLIST atomic arg2 CDATA #IMPLIED>
  <!ATTLIST atomic arg3 CDATA #IMPLIED>
```

The lines starting with ELEMENT indicate the names of elements that may be used within the XML file, and (between round brackets) the sub-elements that an instance of such an element should contain. For example, the first line indicates that a transcript consists of data, general circumstances, and exercises.

More specifically, data (see next 8 ELEMENT lines) consist of demographical information about the participants (in this case the dogs), such as their number (used as a unique identifier), name, breed, and so on. The * indicates that there can be 0 or more training elements within the data element (a dog may have followed multiple types of training), and CDATA indicates that an instance of an element contains "character data" (i.e., any arbitrary combination of characters). Furthermore, an element may have additional information ("attributes"), indicated by !ATTLIST. For example, age is defined in terms of years and months. Here, #REQUIRED and #IMPLIED specify whether the attribute is mandatory or optional.

The next set of lines specifies that general_circumstances consist of background information about the experiment, such as the date and the weather.

Finally, the exercises consist of 0 or more individual exercises, where an individual exercise element is composed of 0 or more times and comment elements, and 0 or 1 (indicated by the ?) exercise_content elements. An exercise also has some attributes to identify the type, trial, and index (i.e., part) of the exercise (where the names separated by the OR-operator | indicate the allowed values). Via times the experimenter can indicate to which times on which camera the part of the transcript corresponds, and comment can be used to specify relevant additional information, such as "part on camera missing". An exercise_content is a nested structure composed of atomic, sequential, and/or parallel events, and comments (where the + operator indicates '1 or more'). Examples of the specific atomic events allowed in our case study are notions like dog_moves_paw, dog_sits, and handler_walks (see the previous section).

3.2 Transcribing Video Material

Based on the ontology introduced in the previous section, observers were asked to transcribe the video material. A partial example is shown below:

```
<exercise_content>
<sequential>
<atomic arg1="handler" arg2="search_forward">commands</atomic>
<atomic arg1="left" arg2="forward">dog_moves_paw</atomic>
<atomic arg1="12" arg2="left" arg3="gallop">dog_moves_to</atomic>
<atomic arg1="down">dog_head_tilt</atomic>
<atomic arg1="1" arg2="2">dog_enters_alley_between</atomic>
<atomic arg1="high">dog_tail_stance</atomic>
<atomic arg1="frequent">dog_wags_tail</atomic>
<atomic arg1="dog" arg2="inside">position_from_alley</atomic>
<atomic arg1="dog" arg2="12">goes_to_pylon</atomic>
<atomic arg1="12">dog_sniffs_at_pylon</atomic>
<atomic arg1="11" arg2="12">dog_leaves_alley_between</atomic>
<atomic arg1="dog" arg2="right">goes_to_pylon</atomic>
<atomic arg1="right">dog_sniffs_at_pylon</atomic>
<atomic arg1="dog" arg2="right">turns_around_pylon</atomic>
<atomic arg1="right" arg2="paw">dog_tilts_pylon_with</atomic>
<atomic arg1="right">dog_looks_inside_pylon</atomic>
<atomic arg1="towards_handler">dog_head_pointed</atomic>
<atomic arg1="11">dog_moves_to</atomic>
<atomic arg1="dog" arg2="left">goes_to_pylon</atomic>
<atomic arg1="dog" arg2="left">turns_around_pylon</atomic>
<atomic arg1="dog" arg2="goody_pylon">goes_to_pylon</atomic>
<atomic arg1="goody_pylon"
    arg2="head">dog_tilts_pylon_with</atomic>
<atomic arg1="goody_pylon">dog_sniffs_at_pylon</atomic>
<atomic arg1="goody_pylon"
    arg2="head">dog_tilts_pylon_with</atomic>
<atomic arg1="goody">dog_eats</atomic>
</sequential>
</exercise_content>
```

For more information about transcribing and the dog experiments, see [1] and the project's website [12].

3.3 Converting Transcripts into Traces

As a next step, transcripts can be analysed in more detail by converting them into formally specified *traces* (i.e., sequences of events over time, cf. [3]), and

checking relevant properties, expressed in the form of temporal logical expressions, against these traces. To convert transcripts into traces, we have written a special piece of software in Prolog. This tool (which is available at [12]) takes a transcript (written in XML) and the corresponding DTD file as input, parses all the events that are described within the transcript, assigns appropriate time intervals to them, and stores the result as a trace. The algorithm used for this simply parses the file from start to end, checks for each line which type of element it represents (e.g., general_circumstances or exercise_content), and converts this to the appropriate syntax using the appropriate Prolog module. For each atomic event, a time step of 10 is taken. The syntax of (an example of) a resulting trace is shown below:

```
atom_trace(dog_number(34), dog_number(34), [range(0, 890, true)]).
atom_trace(dog_moves_to(3,trot), dog_moves_to(3,trot), [range(810,
    820, true), range(710, 720, true), range(650, 670, true), range(360,
    370, true), range(340, 350, true), range(80, 120, true)]).
atom_trace(dog_moves_to(1,trot), dog_moves_to(1,trot), [range(120,
    140, true)]).
atom_trace(dog_moves_to(7,trot), dog_moves_to(7,trot), [range(240,
    270, true), range(140, 150, true)]).
atom_trace(dog_moves_to(12,trot), dog_moves_to(12,trot), [range(730,
    760, true), range(620, 640, true), range(370, 390, true), range(280,
    330, true), range(200, 240, true), range(170, 200, true), range(150,
    170, true)]).
atom_trace(dog_moves_to(6,trot), dog_moves_to(6,trot), [range(850,
    870, true), range(720, 730, true), range(680, 700, true), range(270,
    280, true)]).
atom_trace(dog_moves_to(5,trot), dog_moves_to(5,trot), [range(330,
    340, true)]).
atom_trace(dog_moves_to(9,trot), dog_moves_to(9,trot), [range(820,
    850, true), range(790, 800, true), range(700, 710, true), range(350,
    360, true)]).
atom_trace(dog_moves_to(10,trot), dog_moves_to(10,trot), [range(640,
    650, true)]).
atom_trace(exercise(c,1), exercise(c,1), [range(0, 410, true)]).
atom_trace(exercise(c,2), exercise(c,2), [range(410, 470, true)]).
atom_trace(exercise(c,3), exercise(c,3), [range(470, 890, true)]).
```

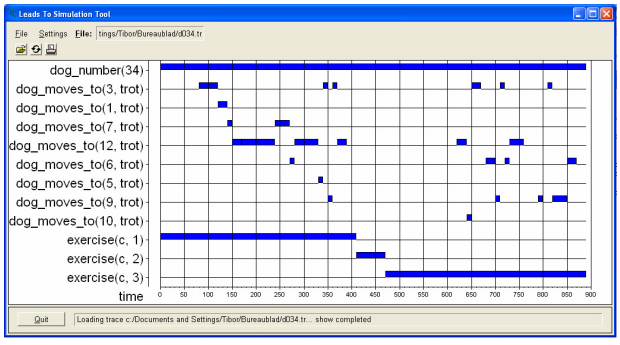

Figure 4. Trace visualisation tool

A trace consists of a number of different *state properties* (i.e., states of the world, indicated by the atom_trace lines), with corresponding time intervals. State properties may represent exercises (e.g., exercise(a,1,rr1)) or events (e.g., dog_moves_paw(right, forward)). For example, the first line indicates that part 'rr1' of exercise A1 lasts from time point 0 to 60.

A graphical user interface [3] visualizes the traces. Figure 4, shows a visualisation of the trace above. In such pictures, time is on the horizontal axis, and the different atoms are on the vertical axis. Blue boxes on top of a line indicate when the atom is true.

3.4 Defining Complex Behaviours in TTL

In order to express properties about complex behaviours in a formal manner, the Temporal Trace Language (TTL) is proposed [3]. This predicate logical language supports formal specification and analysis of dynamic properties, covering both qualitative and quantitative aspects. TTL is built on atoms referring to *states* of the world, *time points* and *traces*, i.e. trajectories of states over time. In addition, *dynamic properties* are temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate $\models$, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) $\models$ p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$.

An example of a dynamic property about dog behaviour would be the following:

for all traces,
if exercise B1-vi lasts from time point tb to te
and at time point t within this exercise,
  the first step of the handler is a step forward with the left leg,
and at time point t2 within this exercise
  the second step of the handler begins,
then there is a time point t1 between t and t2 at which the dog moves

Within TTL, this property is formalised as follows:

P1 ≡ $\forall\gamma$:TRACE tb,te,t,t2:TIME
tb ≤ t < t2 ≤ te &
[[trace_part_of_interest(γ, exercise(b, 1, vi), tb, te)] &
[first_step_of_handler_left_forward(γ, t, tb, te)] &
[second_step_begins(γ, t2, t, te)]]
$\Rightarrow$ $\exists$t1 :TIME [t ≤ t1 < t2 & dog_movement(γ, t1, t2)]

3.5 Checking Complex Behaviours on Traces

To enable the automated verification of dynamic properties specified in TTL against formal traces, a

dedicated tool has been developed. This tool (called the TTL Checker Tool) takes a dynamic property and one or more formal traces as input, and checks whether the dynamic property holds for the traces. Note that these checks can be performed irrespectively of who or what produced the formal traces: humans (or animals), simulators or an implemented (prototype) system. Thus, the TTL Checker Tool can be used to verify properties of both empirical traces, simulated traces and execution traces.

The Checker was implemented in Prolog, and offers a user-friendly graphical editor to create and edit dynamic properties based on tree structures (by means of graphical manipulation and filling in slots, see Figure 5) and the (earlier mentioned) graphical user interface to visualise traces (using XPCE, see Figure 4). A query to check some TTL formula against all loaded traces is compiled into a Prolog clause. Compilation is obtained by mapping conjunctions, disjunctions and negations of TTL formulae to their Prolog equivalents, and by transforming universal quantification into existential quantification. Thereafter, if this Prolog clause succeeds, the corresponding TTL formula holds with respect to all traces under consideration.

Since it does not 'exhaustively' check all possible traces (as e.g. in model checking), the complexity of the checking algorithm is relatively low. It has an upper bound in the order of the product of the sizes of the ranges of all quantified variables. However, a number of specific optimisations make it possible to check realistic dynamic properties with reasonable performance. In practice, the duration of such checks usually varies from one second to a couple of minutes, depending on the complexity of the formula and the traces under consideration. With the increase of the number of traces, the checking time grows linearly. However, it is polynomial in the number of isolated time range variables occurring in the formula under analysis. For more (quantitative) details about the complexity of the checking algorithm, see [3].

The possibility to express dynamic properties in TTL and automatically check them against traces enables the user to formulate complex hypotheses on behaviour (and re-use them). Within a couple of minutes, such hypotheses can be tested against hundreds of traces without having to consider the tapes again. Moreover, when a property fails, the software indicates the cause of the failure, i.e. it pinpoints the exact time point(s) in the traces (and, if necessary, the specific variable instantiations) for which the TTL expression is false. This enables the analyst to inspect that particular part of the trace in more detail by hand.

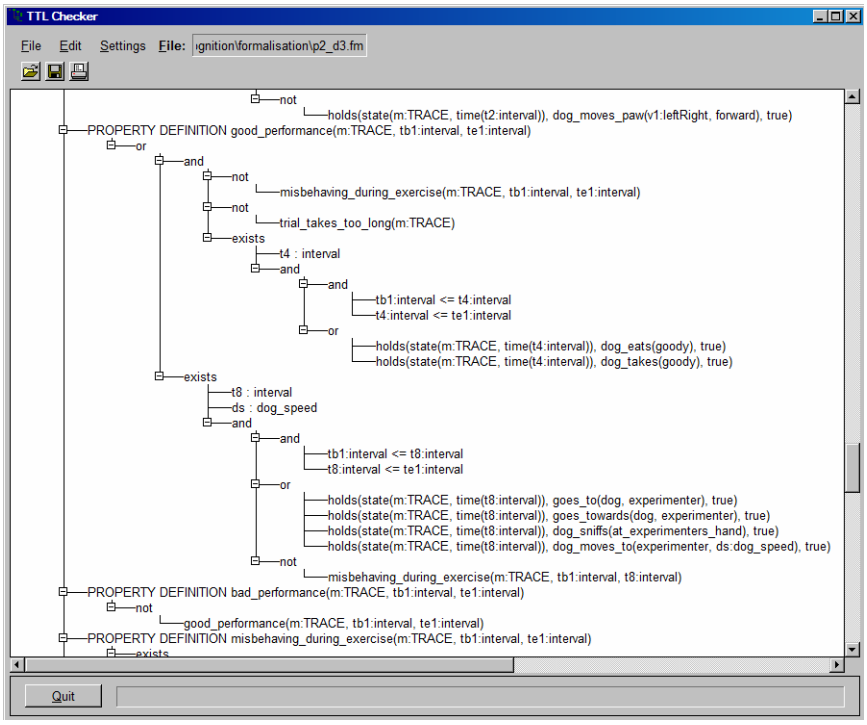

Figure 5. TTL property editor and checker tool

4. Related Work and Conclusions

In experimental research within the social and behavioural sciences, interobserver variability is a known problem. In our research on dog behaviour [1] we see the same phenomenon among the people that help us observe and annotate the videotapes.

In this paper, we propose to reduce interpersonal variability and in the meantime improve reliability and reproducibility by using (novel) formal techniques.

According to the proposed methodology, first the variability with respect to behaviours to be annotated has to be checked in an experiment. For each behaviour, there are two possibilities: basic behaviour (i.e., variability is low and apparently "easy" to observe) and complex behaviour (i.e., variability is high, so more difficult to observe). Complex behaviours then have to be formulated in terms of basic behaviours. Thus, only basic behaviours are to be annotated by the observers. Next, the computer takes the formal definition of complex behaviours and checks whether they hold for the annotated tapes.

The methodology has been illustrated using a case study in dog behaviour. For the example explained in this paper, the complex behaviour is "performance", whereas the basic behaviour is "direction". It was shown that the variability with respect to basic behaviours was high (e.g., 13 out of 30 people characterised a particular case as "good performance", whilst the other 17 characterised it as "bad performance". However, after having the persons define basic behaviour and using these to have the computer judge the performance, only 1 out of 30 participants still characterised the case as "good performance". Thus, the variability with respect to complex behaviours is low, as it depends only on the variability in basic behaviours. The computer does not add variability for the computation of the complex behaviours, which makes it more objective.

In addition, an important use of formal techniques is that researchers later can automatically check other complex behaviours, as long as those behaviours are defined in terms of the basic behaviours already annotated, without anyone having to return to the tapes.

During the case study, the annotation of the tapes has been performed by 6 people, most of which had no background in Artificial Intelligence. Nevertheless, they experienced no problems in working with the formal ontology. The formalisation of the dynamic properties in TTL has been performed by people with a background in Artificial Intelligence, in close collaboration with domain experts.

Concerning related work: we found two papers that also present dedicated tools (other than statistical tools) for automated support in behavioural research [9, 10]. These papers describe a framework that provides support in annotating video material and analysing behavioural data. The current paper elaborates upon these approaches by extending them with a tool to automatically verify complex behaviours in terms of basic behaviours.

5. Acknowledgement

Thanks to W. Akkerman, research assistant, for helping to transcribe videos and to De Jonge Akademie of the KNAW for partially funding the work.

6. References

[1] Alphen, A. van, Bosse, T., Frank, I., Jonker, C.M., and Koeman, F., (2005). Paw preference correlates to task performance in dogs. In: *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, CogSci'05, Lawrence Erlbaum Associates Inc., pp. 2248 – 2253.

[2] Bickman, L., (2000). Validity & social experimentation. Donald Campbell's Legacy, Volume 1. Sage Publications, Thousand Oaks, CA.

[3] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., & Treur, J. (2009). Specification and verification of dynamics in agent models. *Int. Journal of Cooperative Information Systems*. vol. 18, pp. 167 - 193.

[4] Hartmann, D.P., (1977). Considerations in the choice of interobserver reliability estimates. In: *J Appl Behav Anal.*; 10(1): 103–116.

[5] Malek, I.A., Machani, B., Mevcha, A. M., and Hyder, N. H. (2006). Inter-observer reliability and intra-observer reproducibility of the Weber classification of ankle fractures. In: *J Bone Joint Surg Br*; 88-B: 1204 - 1206.

[6] Martin, P., and Bateson, P., (1993). *Measuring Behaviour – An introductory guide*, 2nd edition, Cambridge Univ. Press.

[7] Messick, S. (1989). Validity. In R. L. Linn (Ed.). *Educational Measurement*. 3rd Edition. New York: Macmillan, pp. 13-104.

[8] Myllyluoma, J. and Buck, D. (2008). "Measuring quality in observational data collection" Paper presented at the annual meeting of the American Association For Public Opinion Association, Fontainebleau Resort. Available at URL: http://www.allacademic.com/meta/p17012_index.html.

[9] Noldus, L.P.J.J. (1991). The observer: a software system for collection and analysis of observational data. In: *Behavior research methods, instruments & computers*, vol. 23, no3, pp. 415-429.

[10] Noldus, L.P.J.J., Spink, A.J., and Tegelenbosch, R.A.J. (2002). Computerized video tracking, movement analysis and behaviour recognition in insects. *Comput Electr Agric* 35:201-227.

[11] Wiggins, J. (1973). Personality and prediction: principles of personality assessment. Reading, Mass.: Addison-Wesley.

[12] http://mmi.tudelft.nl/dog-cognition/