Modeling Human Behaviour with Higher Order Logic: Insider Threats

Jaap Boender[§], Marieta Georgieva Ivanova[*], Florian Kammüller[§], Giuseppe Primiero[§]
[*]Technical University of Denmark
firstname.lastname@dtu.dk
[§]Middlesex University London, UK
{initial.lastname}@mdx.ac.uk

***Abstract*—In this paper, we approach the problem of modeling the human component in technical systems with a view on the difference between the use of model and theory in sociology and computer science. One aim of this essay is to show that building of theories and models for sociology can be compared to and implemented in Higher Order Logic. We validate this working hypothesis by revisiting Weber's understanding explanation. We focus on constructive realism in the context of logical explanation. We review Higher Order Logic (HOL) as a foundation for computer science and summarize its use of theories relating it to the sociological process of logical explanation. As a case study on modeling human behaviour, we present the modeling and analysis of insider threats as a Higher Order Logic theory in Isabelle/HOL. We show how each of the three step process of sociological explanation can be seen in our modeling of insider's state, its context within an organisation and the effects on security as outcomes of a theorem proving analysis.**

I. Introduction

Models for cyber security need to incorporate the human factor. We consider these systems as cyber-humane systems. To deal with such systems, computer science needs to tap into sociology. As a step towards utilizing sociological modeling practices for the security modeling of cyber-humane systems, we introduce the method of explanation of sociological processes.

Our aim is to present the way modeling and theory building are used in current sociological practice in order to relate it to formal models for cyber security. To this end, we recapitulate the central concept of 'explanation' used in Weber's understanding sociology [1]. We follow a common introductory textbook for sociologists by Hartmut Esser [2] written in the spirit of Popper's critical rationalism. It offers an approach to understand sociological experiments in a fairly formal way using a logical view on explanation based on the work of the logicians Hempel and Oppenheim [3]. We want to relate this view to the ideas of models, theories and proof in formal methods of computer science to finally be able to critically reflect on current models of human behaviour as used in formal methods of security.

We first review the aforementioned process of explanation in sociology (Section II) and contrast it to formal methods of computer science in particular in Higher Order Logic (Section III). To make such a global endeavor feasible, we narrow down the scope to insider threats to explore the possibilities to model the human factor in a logical model. Insider threats pose an intricate and complex modeling and verification challenge. Reviewing recent advances in data synthesis and analysis for insider threats we propose a tentative higher order logic model for the representation of human behaviour and argue for its feasibility with respect to insider threat analysis (Section IV).

II. Theories and Modeling in Sociology

There is a consensus in all sciences that we strive for truth. Following Popper, it is 'imperative to see and solve the most urgent problems and to solve them by creating true theories' (*wahre Theorien*) [4].[1] Since any theory is always an abstraction from reality, this quest must content itself by approximation of reality. However, reality does not need to manifest itself in the form of the theories but it needs to correspond to the statements in the theory. This conception due to Tarski is known as theory of correspondence [5], [6]. It laid the foundation to model theory in the science of mathematical logic but is also cited as philosophical foundation in mainstream sociology (in particular critical rationalism [2]).

According to Max Weber – one of the forefathers of sociology, the basic process of sociology 'understanding explanation' (*Verstehendes Erklären*) [1] consists of three steps: (a) the 'interpreting understanding' (*Deutendes Verstehen*), where the sociologists needs to understand how the actors interpret their situation, (b) the subjectively meaningful action of the actor, and (c) the effects of the action (see Figure 1).

These three steps incorporate a dimension of interpretation that is unique amongst the sciences as has been observed by Boudon [7]. Alfred Schütz has taken this observation further [8] by coining the notion of *constructions of first order* for the subjective ideas of the actors that determine step (a) of the sociological explanation while he called the models explaining the steps of action (b) and effects (c) as *second order constructions*. Does the aspect of interpretation of the process enforce the use of an unusual logic of explanation? However Weber, Boudon, and Schütz have emphasized that the subjectivity of the social has to be treated with the same objective methods of other sciences [2].

The explanation of sociological phenomena uses a three layered approach following the *logic of explanation* [3] that

[1] 'Rechtfertigung ist kein Ziel; Brillanz und Scharfsinn an sich öde. Wir sollten versuchen, die dringendsten Probleme zu sehen oder zu entdecken und sie durch Aufstellung wahrer Theorien zu lösen.'

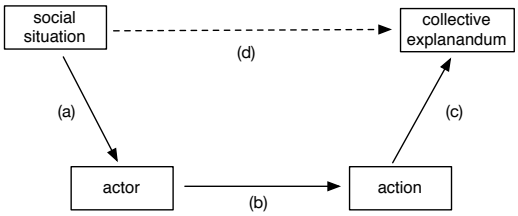

Fig. 1. The 'Grundmodell' of sociological explanation [2]

correspond to Weber's three steps. This approach refines these general three steps as described in Figure 1 by introducing a view on *actors*. The explicit modeling of humans as actors gives rise to distinguish a *macro*-level view from a *micro*-level view. The three steps of Weber's model thus are a macro-micro-macro-transition explaining sociological phenomena by breaking down the global facts from the macro level (a) onto a more refined local view of individual actors at the micro-level (b). Finally those micro-steps are generalized and lifted back on the macro-level (c) to explain the global phenomenon. The formal description of this procedure is described by three transitions between dedicated logics. In the first step, a *situational logic* maps the global context (environment) onto the actor ((a) from the macro to the micro).

The second step in the micro-level of the individual actor (b) is described by a so-called *logic of selection* describing how the actor selects his actions based on the situation (or his perception thereof). The *logic of selection* describes how the actor makes his choice. Examples could be straightforward normative action models in which the actor follows – like an automaton – given rules according to predetermined *norms* or more dynamic forms of action models including, for example, cognitive learning.

The third step called *aggregation logic* comprises the micro-sociological results and lifts them back onto the macro-level to finally explain the social phenomenon that result.

The logic of explanation has been created in 1948 later than Weber's original [9] but it is possible to reconstruct his original hypotheses using this logic. In his analysis of Weber's arguments McClelland [10] casually uses the macro-micro-macro transition, when he reconstructs Weber's explanation of the relationship between 'protestant ethic' and 'the spirit of capitalism'. Protestantism has lead to changes in familial socialization, a 'familial revolution' (macro to micro-level). The change of educational style employed by protestant parents (micro-level) has equipped their children with 'strong internalized achievement drives'. This has created the spirit of capitalism back on the collective, the macro-level, and has lead to the spread of a new type of actor, the entrepreneur.

Correspondence to Computer Security Modeling

The situational logic is usually not modeled in formal models of security. An attacker is assumed to exist and quantified to estimate his strength, but no understanding of motivations, social and psychological factors is attempted. The step from the macro to the micro is a very important change of perspective in many other disciplines like crowd control or car traffic analysis when macro phenomena can be better understood at the micro-level.

The logic of selection can be provided by an action model. The characterization of an attacker's actions and capabilities is the subject of several formal models of security as they are used in security protocol verification. The behaviour of the attacker follows the Dolev-Yao model [11]: the attacker may see every communication, can intercept all messages, and can send anything. In this model, the security of protocols needs to rely exclusively on secrets and cryptographic keys, an assumption that is often considered too strong.

The aggregation logic is a constant factor in formal modeling for security since the only sociological fact we are interested in is security. Nevertheless, the economical aggregation of attacks is of utmost interest, i.e., what are the direct and implicit damages caused by attacks.

From the point of view of modeling systems in logic, it should be noted that the notions of different orders of constructions ([7], [8] see above) seems to correspond to the problem of order in logics. When modeling complex systems, we often need to rely on higher order logics to be able to augment levels of reasoning for applications where we prove properties *of* entities and simultaneously properties *about* entities.

The three steps of the logic of sociological explanation performs a change of perspective between a consideration of the internal psychological and emotional state of individuals and their behaviour in a context as well as the behaviour of groups of individuals. This is reminiscent of the challenges of mathematical explanation that have given rise to the creation of Higher Order Logic to cope with complex logical structures.

III. Higher Order Logic

Bertrand Russell stirred up the world of logics and foundations of mathematics with his paradox. This paradox uses the set $\mathcal{R}$ of all sets not containing themselves. The proposition $\mathcal{R}$ *contains itself* is equivalent to $\mathcal{R}$ *does not contain itself*. Hence, assuming that any statement is either true or false, the definition of $\mathcal{R}$ is inconsistent. However, in naïve set theory this definition is possible. Hence naïve set theory is inconsistent. Russell appears to have discovered his paradox in 1901 while working on his *Principles of Mathematics* [12].

Russell concluded from his observation that mathematics needs a more restrictive foundation. He defined his *Theory of Types*. Ultimately the paradox stems from the application of the Axiom of Comprehension (or Abstraction) stating that any predicate, i.e., propositional function, can be used to define a set. Type theory starts there by imposing a hierarchy on predicates. The lowest level consists of predicates about individuals. The next level consists of predicates about sets of individuals. Then we have the level of predicates about sets of sets of individuals, and so forth leading to an infinite hierarchy. It is then only possible to refer to all objects for which a given

predicate holds if they are all at the same level, or of the same *type*. Hence, a predicate about the set of all sets, like $\mathcal{R}$ would be on a higher level than any level in this hierarchy of types and can thus not be formulated. Russell's Paradox which is based on this set $\mathcal{R}$ is not possible in the theory of types.

However, Russell's type theory has been criticized for being too restrictive and ineffective. In mathematics, the axiomatic system for sets has eventually been refined to overcome Russell's paradox.

The work that revived type theory later is the simple theory of types by Alonzo Church [13]. It is inside the higher order logic (HOL) community considered as the starting point for all classical HOL systems. The simple type theory of Church is similar to the simply typed λ-calculus [14] which is more widely known in computer science. Simple type theory is basically a λ-calculus with simple function types.

The theory of types moved into the world of computer science when logic was used to verify software. Robin Milner introduced the Logic of Computable Functions (LCF) [15] as an implementation for a logic devised by Dana Scott in 1969, but not published until 1993 [16]. The LCF logic is basically domain theory [17] made practical. The use of types seemed appropriate to describe domains and specify functions over them. Milner suggested (at the same time as Hindley) a polymorphic type system [18] for the simply typed λ-calculus. For the support of LCF, in particular to support symbolic evaluation and hence reasoning, the programming language ML was created and later developed in a fully fledged programming language [19]. Out of the attempts to provide reasoning support with ML for the Logic of Computable functions the early Edinburgh LCF system was designed. It has been reengineered by Larry Paulson into the Cambridge LCF system [20]. This system is already an interactive proof support system that helps users construct datatypes and analyze specifications in LCF.

Abstracting from LCF as special application logic, Mike Gordon implemented HOL [21] as a system for interactive verification in ML. A pure implementation of simple type theory interpreted as a logic, the HOL system is a generic tool in the sense that it can be applied to arbitrary logics not just LCF. Picking up this lead, Paulson performed another abstraction step and enhanced the genericity. He created the Isabelle system [22]. The history of the development from LCF to HOL is concisely presented in [23].

Classical HOL

We use the term *classical HOL* to refer to the Higher Order Logic as implemented in the HOL system or the instantiation Isabelle/HOL of the generic tool Isabelle.

As already mentioned in the previous section, HOL is basically a typed λ-calculus. The types include a constructor $\Rightarrow$ to construct the type of functions $\alpha \Rightarrow \beta$ from type α to type β.

This typed λ-calculus is now interpreted as a logic by assuming a type `bool` containing the elements `true` and `false`. Furthermore, there is one infinite type of individuals ι. There are the classical logical junctors, for example $\wedge$ and $\vee$, and an equality = for all types. HOL assumes eight axioms that suffice to lay the foundation for reasoning in the type `bool`, the infinite type and other newly introduced types.

The reason for the *higher order* in its name stems from the intuition that HOL is the bound over a chain of finite-order logics. Starting from first-order logic, continuing with second-order logic, then third-order logic, eventually the chain reaches the so-called ω-order logic. This boundary element is HOL: in first-order logic we can quantify over variables from some simple domains, in second-order logic we can quantify over predicates and functions over such domains. The third step enables quantification over predicates of predicates. In HOL finally, we can reason about arbitrarily nested predicates. Typing is necessary to avoid self-referential predicates.

The expressive strength gained by the unbounded quantification has its price: HOL is neither decidable, nor is it complete. The undecidability of the truth of logical propositions is not surprising because already first-order logic with arithmetic is undecidable. The reason for incompleteness is explained as follows. As reflexivity of equality $x = x$ is one of the axioms of HOL it is necessary to assume that all HOL types have to be nonempty. In order to define a notion of semantic models for HOL it is therefore necessary to introduce standard models containing nonempty domains for the domains of the HOL theory. Adequacy, that is, correctness and completeness, can only be proved with respect to such standard models. Although this might seem like a grave deficiency it is a clear consequence of the expressive power of the logic. Also, it has not yet produced any major limitations.

A consequence of the undecidability of proofs is that reasoning in HOL is interactive. A proposition is stated and its proof is constructed by the user in an interactive process of applying rules until already proved theorems or axioms are reached. In all classical HOL tools the application of rules, theorems, and axioms is transcribed in a so-called proof script. Proofs are not first-class citizens of the logic. Proof objects are neglected as they take up too much space.[2] Once developed theories are rebuilt by rerunning existing proof scripts. This procedure is often referred to as LCF-style as it dates back to the days of LCF-systems.

Although provability is not decidable, the types in classical HOL are decidable. The type system used in HOL resembles strongly the polymorphic higher order type system of ML due to Hindley and Milner. The types of basic formulas are declared in a type environment. For any given type environment the type of a formula may be inferred by the type checker.

Classical HOL has a strict distinction between terms and types. A proposition is a term that inhabits a type. Types are themselves not logical formulas. Still, predicates, i.e., terms, can be used to define new types. In fact this is the main procedure of specifying applications extending the basic HOL.

A *conservative extension* of HOL is an extension of the

[2]There is no foundational reason for this decision. For example, in Isabelle one can now choose to keep proof objects.

existing types of HOL by defining new types from old types, as depicted in Figure 2: a new type σ is defined by copying elements of type τ fulfilling predicate P_σ. The relationship

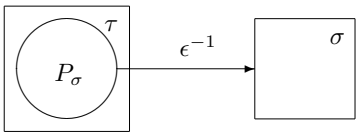

Fig. 2. conservative extension

between the new and the old type is given by an injective embedding ε from the new type σ into τ. Thereby, all properties of elements of the new type σ can be inferred from properties defined over their origins in τ translating them with ε^{-1}. In that way, no new axioms have to be assumed. Additionally, the nonemptiness of the new type σ has to be proved by showing $\exists\, x.\ P_\sigma x$. Consequently, no inconsistencies can be introduced; new types are again nonempty HOL types; the extension is conservative. HOL types are identified by predicates and predicates can only range over the same type – here we see exactly the incarnation of Russell's hierarchy of types.

Correspondence to Modeling in Sociology

Like theory building and modeling in sociology, the HOL modeling seeks correspondence only to reality. The three step process of explanation is very similar to the reasoning of HOL, in that rules are applied in order to explain (or prove) facts from assumptions. However, the way that theories are created is quite converse. The sociological process of explanation uses in each of its three logical steps models from some theory that has its own roots in reality (usually founded by empirical research), for example the theory of control by Hirschi to provide models for the situational logic, and action theories, like decision models, for the logic of selection. The theory building in HOL starts from a minimal set of axioms and two basic types `bool` and ι and builds new theories from there. These are then clearly consistent but their correspondence to domains of reality are in the eye of the beholder. In principle, we can easily define abstract nonsense, for example a new type `world` that is given by one element and is injectively embedded into `bool` by mapping onto `True`. In this world, everything is true but this is nonsensical because there is no falseness.

HOL has no claim to be more than a tool to provide a foundation for mathematics and is like mathematics a completely man-made vehicle consistent in itself (modulo some basic assumptions) to understand (model) what happens in reality to possibly predict or prove (inside its own universe).

However, despite the fact that a logical model in HOL cannot be considered a significant proof of historical existence for the model, it is a consistency proof in line with the notion of verification from computer security. Also, a significant improvement on this notion can be obtained by forcing within the modeling the micro-macro-micro structure from sociology, in particular by including psychological modeling of the actors and their actions.

IV. A Tentative Model of Human Behaviour in HOL

In order to approach a tentative model of human behaviour in HOL, we concentrate on our motivating application area of insider threats. We illustrate how each of Weber's three steps can be supported.

As a running example, we consider first a more data-centric view of insider attacks. Glasser and Lindauer [24] consider the generation of insider threat data using a synthetic data generation framework. The input to the data generation process is largely autonomous and produces intelligent near realistic data. However, the kernel ingredient to this process are basic insider scenarios that are manually inserted. These insider scenarios are constructed using counter-intelligence expert knowledge.

We consider here an example from [24] that nicely shows all three steps of the sociological process of explanation and that thus suits very well as a test case to elicitate requirements to a HOL model for human insider behaviour.

A. Example

'A member of a group decimated by layoffs suffers a drop in job satisfaction. Angry at the company, the employee uploads documents to Dropbox, planning to use them for personal gain.' The data generation process derives so-called 'observables' from this scenario. For the example, the observables are given in the following list [24].

- Data streams end for laid-off co-workers, and they disappear from the LDAP directory.
- As evidenced by logon and logoff times, subject becomes less punctual because of a drop in job satisfaction.
- HTTP logs show document uploads by subject to Dropbox.

When considering building a theory of human behaviour, we can use this attack case since it shows the three steps of Weber's process and can thus serve for requirements elicitation for a comprising HOL model. The situational logic needs to be able to model the process of (a) 'A member of a group decimated by layoffs suffers a drop in job satisfaction.', the logic of selection then must embed (b) 'Angry at the company, the employee uploads documents to Dropbox, planning to ...', and the aggregation logic should express (c) 'use them for personal gain', i.e., effects on the society, for example, damage to the company (workers and clients of company) and wider economical effects. The observables, as given in this example, may be seen as our test cases to see how far a HOL model can encode such characteristic properties. The Dropbox scenario may serve as an illustration for this approach. The full Isabelle/HOL model is contained in the Appendix.

B. Macro to Micro (MaMi)

The transition from the macro-level to the micro-level is determined by the insider's mental characteristics, e.g., psychological state and motivation. The theory supporting this

situational logic needs to characterise insiders. A recent framework for characterising insider threats [25] offers a taxonomy of insider threats based on a thorough survey on results from counterproductive workplace behaviour, e.g., [26], [27] and case studies from the CMU-CERT Insider Threat Guide [28]. The classes identified in this taxonomy are the *Precipitating Event* or catalyst, the individual's *Personality Characteristics*, *Historical Behaviour*, *Psychological State*, *Attitude Towards Work*, *Skill Set*, *Opportunity*, and *Motivation to Attack*.

It is simple to model a taxonomy in HOL since classes are similar to types. We use here the concept of a HOL datatype: datatypes are special HOL-types defined by a finite set of injective constructors provided with good automated reasoning support. As an example, consider the formal representation of *Psychological State* as a datatype.

```
datatype psy_states = happy | depressed | disgruntled
                    | angry | stressed
```

Another example is *Motivation*.

```
datatype motivations = financial | political | revenge
                     | fun | competitive_advantage
                     | power | peer_recognition
```

A practical issue is the integration of causalities, quantification or qualification into this basic model. For example, if an employee is disgruntled this might give rise to a motivation of revenge. In [25], these causalities are expressed by drawing lines between boxes containing the classes of the taxonomy. Such lines express dependencies, like 'motivation for revenge may be caused by anger' but this is not a logical causality, i.e., anger $\Rightarrow$ revenge – a logical causality expresses that anger necessarily implies revenge motivation which might not be the case for all actors.

Logical implication is thus not adequate for expressing dependencies between taxonomy classes. However, HOL offers other constructs like sets, functions and relations that extend the taxonomy classes with a finer grain for modeling dependencies. The values identified for the different classes of the insider threat taxonomy are distinct values by construction since we defined them as the fields of a datatype. However, they may occur in combination. Also we might want to express 'high' or 'low'. This can be easily achieved by building sets of criteria, like the following function that uses the type constructors `set` to define a set of motivations.

```
motivation :: motivations set
```

This construct allows later to attach a range of motivating values to an actor and consequently to use standard HOL-set relations to compare these for qualitative statements, e.g., `motivation_alice` $\subseteq$ `motivation_bob` to express that the motivation to become an insider is higher for Bob. This takes us one step further to a more qualitative model of the insider taxonomy for (non-exclusive) insider criteria like motivation. However, for the `psy_state` datatype, combinations of values, e.g., {`happy, depressed`} may be meaningless and individualized relations like subtyping or inequalities are more useful to introduce a more fine grained qualification and dependencies. In order to add some quantification to each of these factors, it is useful to explicitly model a quantity as part of the assigning function for the actors. The quantity could contain any metrics for a given insider characteristics, e.g., a real number denoting some measure for any of the actors motivational values.

```
quant_motivation :: actor ⇒(real × motivation)
```

The *Precipitating Event* or *Catalyst* has a separate role in the characteristics given in the taxonomy. It can be any event that has the potential to tip the insider over the edge into becoming a threat to their employer. It has been called the 'tipping point' in the literature and can be formalized as a predicate on actors. In order to carry over to the micro-level representation, it is advisable to contain with it the various characteristics about the actor in a combined state.

```
datatype actor_state = State "motivation" "psy_state"
```

Finally, the catalyst is encoded as a tipping point predicate that describes the mutation of an actor to become an insider.

```
definition tipping_point :: actor_state ⇒ bool
tipping_point a ≡ motivation a ≠ {}
                  ∧ happy ≠ psy_states a
```

C. Micro to Micro (MiMi)

At the micro-level, we are interested in modeling the actors including the insider within their context. Therefore, we adopt an approach of modeling the organisation with the actors as a network that can contain various layers of physical, administrative and logical views inspired by Probst and Hansen [29]. This approach originally uses the Klaim calculus to model an organisation and its actors as a graph of locations and actors. The central model is the infrastructure of the organisation, a graph containing locations like rooms but also logical locations like servers. Actors are also modeled as locations in this graph.

```
datatype actor  = Actor string
datatype location = Location nat
datatype node = NA actor | NL location
datatype graph = Graph "(node × node)set"
```

In order to explore insider behaviour in organisational models, we use an abstract view that is inspired by previous work on policy formalisations and analysis [30], [31]. There, invalidation of policies reveals insider attack vectors by model checking system and workflow specifications.

In the current more refined Isabelle/HOL model we express policies over actions `in`, `out`, `move`, and `eval`.

```
datatype action = in | move | eval | out
```

We abstract here from concrete data – actions have no parameters. Policies describe prerequisites for actions to be granted to actors given by pairs of predicates (conditions) and sets of (enabled) actions.

```
type_synonym policy = ((actor ⇒ bool) × action set)
```

We integrate policies with a graph into the infrastructure providing an organisational model where policies reside at

locations and actors are adorned with additional predicates to specify their 'credentials'.

```
datatype infrastructure = Infrastructure
"graph" "location ⇒ policy set" "actor ⇒ bool"
```

These local policies serve to provide a specification of the 'normal' behaviour of actors but are also the starting point for possible attacks on the organisation's assets. The `enables` predicate specifies that an actor `a` can perform an action `a'∈ e` at location `l` in the infrastructure `I` if `a`'s credentials (stored in the tuple space `tspace I a`) imply the location policy's (stored in `delta I l`) condition `p` for `a`.

```
enables I l a a' ≡
 ∃ (p,e) ∈ delta I l. a' ∈ e ∧ (tspace I a ⟶ p(a))
```

The behaviour abstractly specifies that good actors respect the global policy.

```
behaviour I ≡ {(t,a,a'). enables I t a a'}
```

Attacks can be found efficiently by invalidating a global policy and then analysing whether the local policies enable the actors to achieve a state violating this policy. The analysis is part of the aggregation logic.

D. Micro to Macro (MiMa)

The third step of sociological explanation is the aggregation back to the macro level. Since the main effect is that of violating the global corporate policy, the effect to the macro-level is subsumed by the negated global policy: the attack of the insider has the effect that corporate data reaches the outside of the corporate network. We give an outline of the Dropbox analysis finishing the insider explanation with this final step here. The full code of this example application is also part of the Appendix.

The graph representing the infrastructure of the Dropbox case study, contains only the minimal structure, i.e., nodes for the `server`, and the `dropbox`. We only model two actors, the actors `Charly_comp` and `Charly_priv` representing actor Charly once as a member of the company and once as an outsider.

A global policy could be 'no corporate data must leave the corporate network' formalized in our HOL model as follows.

```
global_policy I a ≡  a ∉ company_actors ⟶
    ¬(∃ t. t ∈ company_locations. enables I t a in)
```

Analysis of protocol verification, like the classical Needham-Schroeder public key attack and other insider threat case studies [31] show that a recurring scheme in insider attacks lies in role identification defined as the `UasI` predicate. I.e., the insider manages to play a loyal member and an attacker at the same time.[3]

```
definition UasI ::  "[actor, actor] ⇒ bool"
  where "UasI a b ≡ (act a = act b)"
```

[3]A social engineering attack may be modeled using the scheme as well since then the outside attacker may act like an insider.

Insider attacks link the macro-level insider characterization with behaviours like the above, formalized as `Insider a`.

```
tipping_point (astate a) ⟶ (∀ b. UasI a b)
```

The Dropbox insider threat can now be formalised as an invalidation of the global company policy.

```
⟦ tipping_point (astate (Actor ''Charly_comp''));
  Insider (Actor ''Charly_comp'') ⟧ ⟹
  ¬ global_policy
        dropbox_scenario (Actor ''Charly_comp'')
```

Since `Charly_comp` is at tipping point, he will ignore the global policy and put company data on the Dropbox server. Since `Charly_comp` and `Charly_priv` are the same real person they can share access to the Dropbox directory that is used as a covert channel between the organisation and the outside. `Charly_priv` can get the data. The insider threat can be modeled as an infrastructure example and the attack proved as an Isabelle/HOL theorem.

V. Related Work

The inductive approach to security protocol verification in Isabelle by Paulson [32], the designer of the Isabelle system, picked up on the hype generated by the earlier model checking approach to security by Lowe [33]. In comparison, the inductive approach is more laborious as it requires human interaction, but it is unrivaled in its expressiveness which allows proofs beyond the ones that are usually done in model checkers. The model chosen in the inductive approach can be seen as a micro-level representation of a global communication scenario. Human behaviour is not explicitly addressed nor is any relation to sociology intended. The security protocol defines the 'normal' behaviour of communication partners and the Dolev-Yao model [11] characterizes the attacker's behaviour (sees all, can intercept, and send). The first step of situational logic (interpretation) is implicit in the inductive approach since the attacker has been already introduced. An explicit modeling of this first step needs to integrate a description of how a 'normal' principal turns into an attacker. This is what we additionally address with our first step (MaMi). It is arguable whether the proofs of secrecy of keys in the inductive approach can be seen as a third step (MiMa).

The application of HOL for human behavior modeling has been pioneered by Bella and Coles-Kemp in their formal consideration of *ceremonies* [34]. This concept expands a security protocol to include all those assumptions and other context information that was previously considered as 'out-of band'. Thus, like in our approach, a ceremony also considers the human factor as a central element. The main emphasis in ceremonies, and consequently also in Bella's and Coles-Kemp's work, is on incomplete or partial behaviour of humans, for example, incomplete comparison of values. Bella and Coles-Kemp propose a 'multiple-layer model' consisting of a stack of protocols. The lower part contains the technological protocol layers and the upper part the sociological layers. There is one unique interface (Layer III) where the socio-technological interaction is modeled and analyzed by state

descriptions ('stances') in one direction: from the user to the computer interface. For insider attacks a more complete representation of actors, their internal state as well as their physical and organisational context is needed. Therefore, we model individual actors, their mental state, actions as well as locations that can be physical or logical entities.

Formal techniques for insider analysis have also been applied by Pieters, Dimkov, and Pavlovic [35]. They consider policy alignment to address different levels of abstraction of socio-technical systems. Policy alignment is in their view a refinement of policies to more concrete system representations. Policies are interpreted as first-order logical theories containing all sequences of actions (the behaviours) and expressing the policy as a 'distinguished' prefix-closed predicate in these theories. Refinement of consistent, i.e., policy-fulfilling specifications, is then readily provided by the completeness relation between first-order theories. Although the use of graphs as system representations and local policies attached to those graphs is used in the application example, contrary to our approach Pieters et al. do not use an explicit infrastructure model in their approach [35] but only an abstract formal notion of action sequences to represent systems.

VI. Conclusions

In this paper, we briefly reviewed and summarized sociological explanation according to critical rationalism, introduced and compared to theory building and proof in HOL.

The use of Higher Order Logic permits to express notions of the actor's mental state like motivation or psychological state. At the same time in a same theory the infrastructure of an organisation including the physical network, logical policies and an abstract behaviour of actors in this network can be represented. Thus, human behaviour can be modeled for insider threats according to Weber's three steps of sociological explanation. Higher Order Logic allows formalisation of actor's properties for all three steps as we illustrated on a case study of the Dropbox insider attack.

References

[1] Weber, M.: Wirtschaft und Gesellschaft. Grundriss der verstehenden Soziologie. Tübingen (1972) 5. Auflage.

[2] Esser, H.: Soziologie – Allgemeine Grundlagen. Campus (1993)

[3] Hempel, C.G., Oppenheim, P.: Studies in the logic of explanation. Philosophy of Science **15** (April 1948) 135–175

[4] Popper, K.R.: Objektive Erkenntnis. Hoffmann und Campe (1993)

[5] Tarski, A.: Der wahrheitsbegriff in den formalizierten sprachen. Studia Philosophica **1** (1935) 261–405

[6] Tarski, A.: The semantic conception of truth: and the foundations of semantics. Philosophy and Phenomenological Research **4** (March 1944)

[7] Boudon, R.: Die Logik des gesellschaftlichen Handelns. Eine Einführung in die soziologische Denk- und Arbeitsweise. Darmstadt und Neuwied (1980)

[8] Schütz, A.: Begriffs- und theoriebildung in den sozialwissenschaften. In: Gesammelte Aufsätze, Band 1: Das Problem der sozialen Wirklichkeit. Springer Netherlands (1971) 55–76

[9] Weber, M.: Die protestantische Ethik und der Geist des Kapitalismus. In: Max Weber, Gesammelte Aufsätze zur Religionssoziologie, Tübingen (1978) 7. edition, (first edition 1920).

[10] McClelland, D.C.: The achieving society. Van Nostrand, Princeton, NJ (1961)

[11] Dolev, D., Yao, A.C.: On the security of public key protocols. In: 22nd Annual Symposium on Foundations of Computer Science, SFCS '81, IEEE (1981)

[12] Russell, B.: Principles of Mathematics. Cambridge University Press (1903)

[13] Church, A.: A simple theory of types. Journal of Symbolic Logic (1940) pages 56–68

[14] Pierce, B.: Types and Programming Languages. Wiley (2002)

[15] Milner, R.: Logic for computable functions; description of a machine implementation. Technical Report STAN-CS-72-288, A.I. Memo 169, Stanford University (1972)

[16] Scott, D.: A type-theoretical alternative to iswim, cuch, ohwy. Theoretical Computer Science **121** (1993) 411–440 Annotated version of the 1969 manuscript.

[17] Scott, D.: Domains for Denotational Semantics. Volume 140 of Lecture Notes in Computer Science. Springer (1982)

[18] Milner, R.: A theory of type polymorphism in programming. Journal of Computer and System Science **17**(3) (1978) 348–375

[19] Paulson, L.C.: ML for the Working Programmer. Cambridge University Press (1991) Second edition 1996.

[20] Paulson, L.C.: Logic and Computation: Interactive proof with Cambridge LCF. Cambridge University Press (1987)

[21] Gordon, M.J.C., Melham, T.F.: Introduction to HOL, a Theorem Proving Environment for Higher Order Logic. Cambridge University Press (1993)

[22] Paulson, L.C.: Isabelle: A Generic Theorem Prover. Volume 828 of LNCS. Springer (1994)

[23] Gordon, M.J.C.: From LCF to HOL: a short history. In G. Plotkin, Colin P. Stirling, M.T., ed.: Proof, Language, and Interaction. MIT Press (2000)

[24] Glasser, J., Lindauer, B.: Bridging the gap: A pragmatic approach to generating insider threat data. [36]

[25] Nurse, J.R.C., Buckley, O., Legg, P.A., Goldsmith, M., Creese, S., Wright, G.R.T., Whitty, M.: Understanding Insider Threat: A Framework for Characterising Attacks. [37]

[26] Martinko, M.J., Grundlach, M.J., Douglas, S.C.: Toward an integrative theory of counterproductive workplace behaviour. International Journal of Selection and Assessment **10**(1–2) (2002) 36–50

[27] Marcu, B., Schuler, H.: Antecedents of counterproductive behaviour at work: a general perspective. Journal of Applied Psychology **89**(4) (2004) 647

[28] Cappelli, D.M., Moore, A.P., Trzeciak, R.F.: The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud). 1 edn. SEI Series in Software Engineering. Addison-Wesley Professional (February 2012)

[29] Probst, C.W., Hansen, R.R.: An extensible analysable system model. Information Security Technical Report **13** (2008) 235–246

[30] Kammüller, F., Probst, C.W.: Invalidating policies using structural information. [36]

[31] Kammüller, F., Probst, C.W.: Combining generated data models with formal invalidation for insider threat analysis. [37]

[32] Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security **6**(1-2) (1998) 85–128

[33] Lowe, G.: Casper: A compiler for the analysis of security protocols. In: Computer Security Foundations Workshop (CSFW '97). (1997)

[34] Bella, G., Coles-Kemp, L.: Layered analysis of security ceremonies. In Gritzalis, D., Furnell, S., Theoharidou, M., eds.: SEC. Volume 376 of IFIP Advances in Information and Communication Technology., Springer (2012) 273–286

[35] Pieters, W., Dimkov, T., Pavlovic, D.: Security policy alignment: A formal approach. IEEE Systems Journal **7**(2) (2013) 275–287

[36] Proceedings of the second IEEE Workshop on Research in Insider Threats, WRIT'13. In: WRIT'13, IEEE (2013)

[37] Proceedings of the third IEEE Workshop on Research in Insider Threats, WRIT'14. In: WRIT'14, IEEE (2014)

VII. Appendix

```
theory enables
imports Main
begin
datatype action = in | move | eval |out
datatype actor  = Actor string
type_synonym policy = "((actor ⇒ bool) × action set)"
datatype location = Location nat
datatype node = NA actor | NL location
datatype graph = Graph "(node × node)set"
datatype infrastructure = Infrastructure "graph" "location ⇒ policy set"  "actor ⇒ bool"
primrec act :: "actor ⇒ string"  where  "act(Actor n) = n"
primrec loc :: "location ⇒ nat"  where  "loc(Location n) = n"
primrec gra :: "graph ⇒ (node × node)set"  where  "gra(Graph g) = g"
definition nodes_graph :: "graph ⇒ node set"  where "nodes_graph g ≡ { x. ∃ y. (x,y) ∈ gra g) ∨ (y,x) ∈ gra g }"
definition actors_graph :: "graph ⇒ actor set"
  where  "actors_graph g ≡ {x. ∃ y. NA(Actor y) : nodes_graph g ∧ x = Actor y}"
definition locs_graph :: "graph ⇒ location set"
  where  "locs_graph g ≡ {x. ∃ y. NL(Location y) ∈ nodes_graph g ∧ x =  Location y}"
primrec graphI :: "infrastructure ⇒ graph" where "graphI (Infrastructure g d c) = g"
primrec delta :: "[infrastructure, location] ⇒ policy set"  where "delta (Infrastructure g d c) = d"
primrec tspace :: "[infrastructure, actor] ⇒ bool" where "tspace (Infrastructure g d c) = c"
primrec actorsI :: "infrastructure ⇒ actor set"  where "actorsI (Infrastructure g d c) = actors_graph g"
datatype psy_states = happy | depressed | disgruntled | angry | stressed
datatype motivations = financial | political | revenge | curious | competitive_advantage | power | peer_recognition
datatype actor_state = Actor_state "psy_states" "motivations set"
primrec motivation :: "actor_state ⇒ motivations set" where "motivation  (Actor_state p m) =  m"
primrec psy_state :: "actor_state ⇒ psy_states"  where "psy_state  (Actor_state p m) = p"

definition tipping_point :: "actor_state ⇒ bool" where
  "tipping_point a ≡ ((motivation a ≠ {}) ∧ (happy ≠ psy_state a))"
definition UasI ::  "[actor, actor] ⇒ bool" where "UasI a b ≡ (act a = act b)"
consts astate :: "actor ⇒ actor_state"
definition Insider :: "[actor] ⇒ bool"
  where "Insider a ≡ (tipping_point (astate a) ⟶ (∀ b. UasI a b))"

definition enables :: "[infrastructure, location, actor, action] ⇒ bool"
  where  "enables I l a a' ≡  ∃ (p,e) ∈ delta I l. a' ∈ e ∧ (tspace I a ⟶ p(a))
definition behaviour :: "infrastructure ⇒ (location × actor × action)set"
  where "behaviour I ≡ {(t,a,a'). enables I t a a'}"
definition ID :: "[actor, string] ⇒ bool"   where "ID a s ≡ (act a = s)"
consts role :: "actor × string ⇒ bool"

(* Dropbox example; infrastructure 1 = company server, 2 = dropbox *)
definition company_actors :: "actor set" where "company_actors ≡ Actor ''Charly_comp''"
definition company_locations :: "location set" where "company_locations ≡ Location 1"

definition global_policy :: "[infrastructure, actor] ⇒ bool"
  where "global_policy I a ≡ a ∉ company_actors ⟶ ¬(∃ t ∈ company_locations. enables I t a in)"

definition ex_graph :: "graph"
  where "ex_graph ≡ Graph {(NA (Actor ''Charly_comp''), NL (Location 1)),
      (NL (Location 2), NL (Location 1)), (NA (Actor ''Charly_priv''), NL (Location 2))}"
definition local_policies :: "location ⟶ policy set" where
  "local_policies ≡ (λ x. if x = Location 1 then {(λ x. role (x, ''employee''), in,out)}
                  else (if x = Location 2 then
                            {((λ x. (ID x ''Charly_priv'') ∨ (ID x ''Charly_priv'')), in,out)} else {}))"
definition ex_creds :: "actor ⇒ bool"
  where "ex_creds ≡ (λ x. if x = Actor ''Charly_comp'' then
                        role(Actor ''Charly_comp'',''employee'') else False)"
definition dropbox_scenario :: "infrastructure"
  where "dropbox_scenario ≡ Infrastructure ex_graph local_policies ex_creds"

lemma ex_inv : "⟦ tipping_point (astate (Actor ''Charly_comp'')); Insider (Actor ''Charly_comp'') ⟧
               ⟹  ¬ global_policy dropbox_scenario (Actor ''Charly_comp'')"
apply (simp add: dropbox_scenario_def global_policy_def)
apply (subgoal_tac " Actor ''Charly_comp'' =  Actor ''Charly_priv''")
apply (rule conjI)
apply (simp add: company_actors_def)
apply (rule_tac x = "Location 1" in bexI)
apply (simp add: dropbox_scenario_def ex_creds_def local_policies_def ex_graph_def)
apply (simp add: company_locations_def)
apply (unfold Insider_def)
apply ((drule mp), assumption)
apply (drule_tac  x = "Actor ''Charly_priv''" in spec)
by (simp add: UasI_def)
end
```