1. **Project Management**

The **VVSimLab** project is designed to explore the process of verification and validation of models and simulations by treating all models as uncertain. The basic concept of **VVSimLab** is the **project**. Projects are handled by functions in `toplevel.w`.

The models are defined in a **general systems** manner. We can assume that any of the standard system definitions facilities can be used.

A **system** is an **object** with **behavior**. This definition represents a recursive definition. All systems operate in an **environment**, although some may ignore their environment. The **system assumption** is that the system can be behaviorly separated from its environment. By behavior we mean the time response of a system to inputs from its environment. At its simplest level, behavior is a set of **tuples** based on defined **quantities**.

The **VVSimLab** project is an experimental programming environment to explore (1) modeling, (2) simulation development, and (3) V&V procedures derived from (1) and (2). The basic approach is to use **general systems theory** as proposed in the 1970s in Klir's books as the highest order schemata. The models will be derived **declaratively** and then processed with a **controlled vocabulary** to produce a simulation. This same approach will be used to develop the V&V model.

The **VVSimLab** project is coded in Prolog and gForth. The code is developed in separate files designated to support each of the attributes named in the GST section.

2. **The Role of Language in GST**

   (1) The fundamental issue is that scientific and engineering progress relies on the written word.
   
   (2) Science and engineering is done in communities.
   
   (3) These communities develop specialized vocabulary concerning the systems of interest.
   
   (4) The purpose of GST is re-integrate these vocabularies so that the general systems practitioner sees the system in every vocabulary.

3. **Desiderata for System Definitions: Traits**

   The fundamental concepts for systems definitions is given here and taken from Klir[?]. For completeness, a trait is “[6. a.] A particular feature of mind or character; a distinguishing quality; a characteristic; spec. of a culture or social group” (Oxford English Dictionary, 2d Edition, 1989, on-line). Or, more specifically, a trait is a characteristic or distinguishing quality of a system.

   However, traits can only be ascribed to things that exist, or what we call entities. An entity is “[s]omething that has a real existence; an ens, as distinguished from a mere function, attribute, relation, etc” (OED). Hence we are interested in measured **traits of entities**. The phrases ontological and ontological import relate to existence. On the flip side, ens rationis is a thing which has an existence only as an object of reason.

   We can further break traits down into essential attributes and non-essential attributes. We reserve the term attribute to mean essential attribute. Trait retains its meaning.

   Quantities are entities that at every time instance have a definite value and on the basis of which a variable is measured. [?] seems to distinguish between quantity as
“the sine qua non of a variable” and variable to take on its logical or mathematical nature: a signification but not ontological. Values are types, essentially. But we’re going to add in some of the logical necessities such as units.

Frames are collections of bindings denoted in print as pairs \((s \mapsto d)\), where \(s\) is a symbol and \(d\) is a definition. \(\mapsto\) ("bowtie") is a macro operation, i.e., pure substitution (An instruction written so as to be equivalent to a chosen set of several instructions. OED). Unification is the fundamental pattern-matching operation. There is always (express or implied) evaluation function. Frames are the method of expressing objects, be they ontological or otherwise.

4. Desiderata on System Properties

The characteristics of a system definition are, according to Klir,

\(D1\): Based on constant traits. Constant traits do not change character among system instances.

\(D2\): Based on characteristic traits hypothesized as completely known

\(D3\): Is not under-determined. That is, Based on traits which make it possible to determine uniquely for each secondary trait whether or not it is consistent with the given traits.

\(D4\): Is not over-determined. That is, it Has no redundant traits.

Clear \(D3\) and \(D4\) are not always possible, especially during early research.

4.1. The Fundamentals of Definition. So far we have entities and traits on the one hand with frames and \(\mapsto\) with \(\{\}\) for representations.

5. Referent Definitions

Clearly, terms must have referents: a term must have meaning with respect to some objects or concepts already understood. In physics class, we called this a free-body diagram, with the idea that we produced a diagram that only indicated the system traits needed for the problem at hand. But there is another way to think about this.

A fundamental concept of VVSimLab is that what we really learn in our education are schemata (sing. schema). For example, in mechanics there are six fundamental simple machines (schemata) and all machines can be made from those six\(^1\) Understanding these simple machines is as important to machine designers as understanding elements of a programming language is to programmers.

5.1. Defining Frames. The named frame is the fundamental concept in the model. We call it a named frame to distinguish it from the ubiquitous use of object in the modeling and simulation literature. The frame support is so important that it is developed here.

\(^1\)Traditionally, inclined plane, screw, wedge, lever, wheel and axle, pulley.
5.1.1. Create Frame. A frame is really just a KB entry based on the type of element desired. This is so that gensym to help disperse the references.

5.1.2. Reading a Frame. Since the basic idea is to have a triple name(attr, val), reading a frame is faster in the code directly. However, a general case is available here.
4

\[
\text{(read-frame 4a) } \equiv \\
\text{read-frame(Name, Attribute, Value} : - \text{atom(Name),}
\text{atom(Attribute),}
A =.. [\text{Name,Attribute,Value}],
A.
\text{read-frame(Name,Attribute,_) : -}
\text{print(Name),}
\text{print(' and '),}
\text{print(Attribute),}
\text{print(' not found.'),}
nl,
!,fail.
\]

\text{Macro referenced in scrap 3a.}

5.1.3. Writing a Frame. Writing a frame is easier as a call.

\[
\text{(write-frame 4b) } \equiv \\
\text{write-frame(Name, Attribute, Value} : - \text{atom(Name),}
\text{atom(Attribute),}
\text{nonvar(Value),}
A =.. [\text{Name,Attribute,}],
\text{(retract(A);true),}
B =.. [\text{Name,Attribute,Value}],
\text{assert(B).}
\text{write-frame(Name,Attribute,Value} : - \text{print(Name),}
\text{print(', '),}
\text{print(Attribute),}
\text{print(', and '),}
\text{print(Value),}
\text{print(' fails.'),}
!,fail.
\]

\text{Macro referenced in scrap 3a.}

5.1.4. Destroy Frame Selector. This guarantees that a frame selector is removed.

\[
\text{(destroy-frame-selector 4c) } \equiv \\
\text{destroy-frame-selector(Name, Attribute} : -
\text{A =.. [Name,Attribute,],}
\text{retractall(A).}
\]

\text{Macro referenced in scrap 3a.}

5.1.5. Destroy Frame. This effectively removes the reference.
\[
\langle \text{destroy-frame } 5 \rangle \equiv \\
\text{destroy-frame}(	ext{Name}) :- \\
\text{A =.. [Name, _, _],} \\
\text{retractall(A),} \\
\text{B =.. [Name, Type],} \\
\text{retract(B),} \\
\text{C =.. [Type,Name],} \\
\text{retract(C).}
\]

Macro referenced in scrap 3a.

6. FUNDAMENTAL TRAITS OF SYSTEMS

T1: The set of external quantities: values as observations of quantities. externalquant.pl
T2: Resolution level. resolution.w
T3: Activity (observation of all quantities)
T4: Behavior
T5: Set of States
T6: Set of Transitions of states
T7: Instantaneous state
T8: Program "variable part of the system" (primarily states)
T9: Structure of system "constant part of the system"
T10: Universe of discourse
T11: Couplings
T12: CC
T13: ST

7. FUNDAMENTAL PROBLEMS

(1) White box: can see internals
(2) Analysis
(3) Synthesis
(4) Black box

8. FUNDAMENTAL SYSTEMS DEFINITIONS

(1) The set of external quantities.
(2) Resolution level
(3) A given activity (observation of all quantities)
(4) Permanent Behavior
(5) Complete set of States
(6) Complete set of Transitions of states
(7) Universe of discourse including complete behavior of all components.
(8) Real Couplings
(9) Real CC
(10) Real ST
(11) Structure of system "constant part of the system"

9. THE ROLE OF LANGUAGE

Duh! Abstraction vocabularies!
10. Implementation

10.1. Knowledge Base. The fundamental concept is the named frame, which is equivalent to a named object. The name is not necessarily the name used by the developer; it can be gensym. Such a mechanism takes maximum advantage of Prolog. The basic schema is

\[ \text{name}(\text{type}, \text{selectlabel}, \text{value}) \].

Developer names are kept separate from the frame base.

10.2. Constituent Definitions.

"vvsimlab.pl" \equiv

\[
\begin{align*}
vvsimlab_load :&- \\
&\quad \text{consult(toplevel)}, \\
&\quad \text{consult(externalquant)}, \\
&\quad \text{consult(resolution)}.
\end{align*}
\]

\langle frame-processes 3a \rangle
\langle quick-create-test 3c \rangle
\diamond