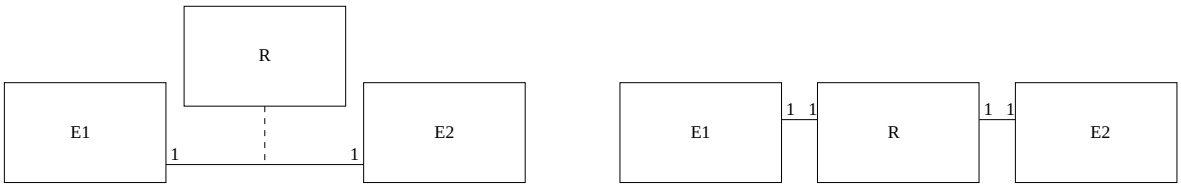
Formal semantics of ERDs

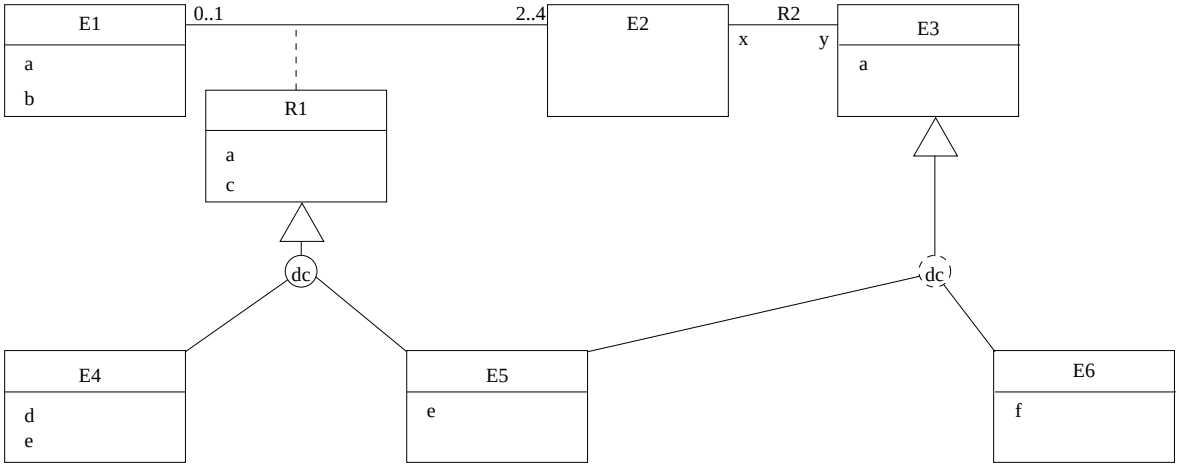
Maarten Fokkinga, Verion of 4th of Sept, 2001

The meaning of an ERD. The notations introduced in the book *Design Methods for Reactive Systems* [2] are meant to describe part of reality, namely the subject domain of the SuD. So, the principal interpretation of the notational ingredients consists of individuals, concepts, and properties in the real world; the interpretation is inherently informal and subjective! Nevertheless, the notation also has an interpretation of its own, in abstract mathematical terms rather than concrete real world terms. The existence of such an interpretation is important, because it enables us to decide various questions about the notation without resorting to informal (hence subjective) arguments. For example, without knowing the intended, principal interpretation into the real world, we can decide the equivalence of the following two ERDs:



In this note we give a "translation" of an ERD to its formal semantics. It will turn out that the formal semantics takes a lot more symbols and piece of paper than the ERD itself. This indicates that ERDs are a concise notation; giving only the mathematical formulas instead of a graphical ERD would be unpractical. The mathematical semantics comes into play only to give rigorous proofs of theorems about ERDs (such as verifications of *checking and manipulations by tools*!), or to explain dark corners of the principal interpretation of ERDs.

Example ERD. We shall not *formally* translate ERDs into the mathematical semantics, but instead translate *by hand* one typical example. Since the mathematical semantics only depends on the ERD and not on the intended, principal interpretation, we have rather schematic names. Here is the example:



Notice that there occur entities, relations, attributes (whose names are local to the entity in which they occur), roles, various cardinality properties, an association relation, two multiple specializations, one multiple generalization, and static and dynamic dc-properties. So, much of the ERD notation does occur in the example. As an example illustration of the use of the

mathematical semantics, we shall derive a surprising property in the corollary below.

Abstracting from reality. Before delving into details, let us first consider the informal but principal interpretation of the given ERD and try and abstract from real world aspect and replace them by mathematical artifacts instead.

The ERD is about extensions and extents, and attributes and roles. Extensions and extents are *sets* of real world *identities*. We shall abstract from real world identities, and postulate a set *Id* instead (whose members we call identities, of course). We shall also abstract from the *values* that attributes can take, and postulate a set *Val* of values. A role is nothing but an identity-valued attribute; no new concepts are involved here. So far for the real world concepts involved in the principal interpretation of the ERD; in the formal semantics, the identities and values are postulated, and then extensions and extents are sets of identities just as in the real world, and attributes and roles are functions, mapping identities to values and identities, respectively, just as in the real world.

The core of the interpretation of the ERD is a characterization of the possible states of the real world. For example, according to the dynamic specialization part in the ERD, each currently existing $E3$ instance is either an $E5$ instance or an $E6$ instance. Thus the ERD characterizes possible states of the world — as far as extensions, extents, attribute values and roles are concerned. The formal semantics will do the same; after the (very simple!) postulations of *Id* and *Val* it gives one (huge!) formula, '*State*', telling on the one hand what components a state has (27 components for the example ERD, as we shall see), and on the other hand what properties the components have (in order to be a proper state).

To be complete, the formal semantics must also define the way in which the state may change. For example, may it happen that an identity in one extension becomes a member in another extension when the state changes? Therefore, the formal semantics also gives a property, 'Δ*State*', for *pairs* of states, that characterizes possible state changes. In this characterization we may also include additional *dynamic properties* that cannot be expressed in conventional ERD notation.

Thus the formal semantics consists of postulations of *Id* and *Val*, and the two formulas *State* and Δ*State*.

Our notation. The formal semantics we give is completely formulated in terms of conventional set theory and predicate logic. We shall use the Z notation [1] for this purpose. Apart from being a quite systematic notation for sets and functions (containing some unconventional but convenient squiggles, and accompanied with several tools such as type-checking and theorem proving), the Z notation also contains the *schema* notation. A schema is nothing but an x-tuple of things, together with constraining properties; *State* and Δ*State* will be schema's. The related schema manipulations facilitate a far going modularization of the formulas. Modularity means that the author can group together precisely those (sub)formulas that he considers to belong together. Using modularity, we can present *State* and Δ*State* in *understandable* little bits (little schema's), while being *formal* at the same time (so that type-checking and other tools can be applied)!

In order to make the mathematical constructions very clear, and in order not to be distracted by the Z specific schema manipulations, we first present each of *State* and Δ*State* as one huge schema. We call this the monolithic formulation. Later on we will show how a modular approach looks like.

The monolithic formulation

Postulation. There is one big universe Id of identities, and one set Val of attribute values:

$$[Id, Val]$$

Thus Id and Val are sets of which we do not know anything except that they exist; regarding type checking of the Z notation they function as a new basic types. However, an identity of a relation should be a *pair* of identities, or at least should be interpretable as such. Therefore we specify that pairs of identities can be identified with single identities via an *injective* function, called *pair*:

$$pair : Id \times Id \rightarrowtail Id$$

State. We give the characterization of 'state' in one huge schema. The upper part contains 27 declarations, giving the 27 components of a state. These are, in order, the extensions and extents (subsets of Id) for all the entities and relations, and the attribute and role functions:

State

$E1\_exn,\ \ldots,\ E6\_exn,\ R1\_exn,\ R2\_exn\ : \mathbb{P}\, Id$	8 extensions
$E1\_ext,\ \ldots,\ E6\_ext,\ R1\_ext,\ R2\_ext\ : \mathbb{P}\, Id$	8 extents
$a\_E1,\ a\_R1,\ \ldots,\ f\_E6\ : Id \nrightarrow Val$	9 attributes
$x\_E3,\ y\_E2\ : Id \nrightarrow Id$	2 roles
properties	

Notice that attribute function a_E1 is declared to be a partial function from Id to Val, although its domain is precisely $E1\_ext$. The reason for doing so is that, in Z, the scope of the declarations is *only* the lower part of the schema, so we cannot declare in the upper part that a_E1 has type $E1\_ext \rightarrow Val$. Instead we shall add such properties in the lower part of the schema.

The lower part of the schema contains the properties that should hold of the components in order that they form a proper 'state'. We proceed in arbitrary order. (In the modular approach we would definitely bring in more structure into this set of properties.) First we have that the extents form a subset of the extensions:

$$E1\_ext \subseteq E1\_exn;\ \ \ldots;\ \ E6\_ext \subseteq E6\_exn;\quad R1\_ext \subseteq R1\_exn;\ \ R2\_ext \subseteq R2\_exn$$

Secondly, the relation extensions are *isomorphic* to the Cartesian products of the participating entity extensions:

$$pair \in E1\_exn \times E2\_exn \rightarrowtail\!\!\!\!\!\rightarrow R1\_exn$$
$$pair \in E2\_exn \times E3\_exn \rightarrowtail\!\!\!\!\!\rightarrow R2\_exn$$

So, if $e1, e2$ are identities for $E1, E2$, then $pair(e1, e2)$ is the identity in $R1\_exn$ representing the fact that $e1$ and $e2$ are related by $R1$. The multiplicity properties further restrict the relation extents:

$$\forall\, e1 : E1\_ext \bullet \#\{e2 : E2\_ext \mid pair(e1, e2) \in R1\_ext\} \in 2\,..\,4$$
$$\forall\, e2 : E2\_ext \bullet \#\{e1 : E1\_ext \mid pair(e1, e2) \in R1\_ext\} \in 0\,..\,1$$

Next, the attribute functions have the right *domain*, and the role functions are completely fixed:

$$a\_E1 \in E1\_ext \to Val; \quad \ldots; \quad f\_E6 \in E6\_ext \to Val$$
$$x\_E3 = (\lambda\, e3 : E3\_ext \bullet \{e2 : E2\_ext \mid pair(e2, e3) \in R2\_ext\})$$
$$y\_E2 = (\lambda\, e2 : E2\_ext \bullet \{e3 : E3\_ext \mid pair(e2, e3) \in R2\_ext\})$$

If the ERD had also given the types of the attributes, we could have specified the *ranges* or more properties of the attribute functions here as well. Below we will see that attribute functions of a supertype are also applicable to their subtypes (inheritance). The attribute functions e in $E4$ and $E5$ are completely unrelated.

Finally, we formulate the properties expressed by the static and dynamic dc-specialization:

$$\langle E4\_exn, E5\_exn\rangle \text{ partitions } R1\_exn \;\wedge\; E4\_ext \cup E5\_ext \subseteq R1\_ext$$
$$E5\_exn = E6\_exn = E3\_exn \;\wedge\; \langle E5\_ext,\, E6\_ext\rangle \text{ partitions } E3\_ext$$

It follows that $E4\_ext \subseteq R1\_ext = \text{dom}\, a\_R1$, so a_R1 is applicable to elements from $E4\_ext$ as well: *inheritance*.

This completes schema *State*.

Corollary. To show the use of the formal semantics in order to explore dark corners of the informal ERD semantics, we give here a purely logical (mathematical, formal) reasoning within *State* and interpret the outcome back into reality, with a surprising result.

Within the lower part of *State* it follows from the properties dealing with specialization that $E3\_exn = E5\_exn \subseteq R1\_exn$. Moreover, there is also a formula that $R1$ is a relation between $E1$ and $E2$, namely $pair \in E1\_exn \times E2\_exn \rightarrowtail\!\!\!\!\!\twoheadrightarrow R1\_exn$. From these two properties it follows that $E3\_exn$ is isomorphic to a subset of $E1\_exn \times E2\_exn$, so that we may say that $E3$ is a *specialization of an association relation between* $E1$ and $E2$. Apparently the author of the ERD has forgotten to draw that in the diagram, or considered that information not worth to be drawn! Or he has made a mistake, brought to light by the formal semantics...

Opmerking. Tot nu toe heb ik altijd gedacht dat extensions disjunct zijn tenzij sprake is van een specialisatie (zoals bij $E4$, $E5$, $E6$). Dat zou voor het voorbeeld ERD betekenen:

$$\text{disjoint}\langle E1\_exn, E2\_exn, E3\_exn, R1\_exn\rangle$$

Maar bij nader inzien blijkt zo'n disjointness nergens genoemd te worden in Roel's boek, en is bovenstaande eigenschap inconsistent met de eigenschap $E3\_exn \subseteq R1\_exn$ die elders in *State* afleidbaar is. Toch handig, om een formele semantiek te hebben die dit soort problemen/misverstanden bespreekbaar maakt.

Opmerking. Als we geformuleerd hadden dat relatie-extensies niet slechts *isomorf* zijn met, maar zelfs *gelijk* zijn aan cartesische producten, dan hadden we ook '$E4\_exn, E5\_exn, E3\_exn : \mathbb{P}\, Id$' moeten wijzigen in de eigenschap dat $E4\_exn, E5\_exn, E3\_exn$ deelverzameling zijn van cartesische producten; want anders resulteert er een inconsistentie. De semantiek die wij nu geven (met isomorfie) stelt ons in staat om "blindelings" te werk te gaan en de conjunctie te nemen van alle eigenschappen die we stipuleren (zonder sommige te moeten herroepen).

State change. Now we want to formulate which pairs of states form a valid change in the world described by the given ERD. Denoting the components of an "old" state by exactly the same identifiers as in schema *State*, and denoting the components of a "new" state by the corresponding primed identifiers, the formula characterising the valid state changes has the following form (where '$\Delta State$' is a single identifier):

$\Delta State$

all declarations of schema *State*
all declarations of schema *State* with a prime at each declared identifier

---

all properties of schema *State*
all properties of schema *State* with a prime at each declared identifier
additional properties

This is a huge schema indeed. Making a modest use of the schema notations of Z, we can write a much shorter but equivalent schema as follows:

$\Delta State$

$State;\ \ State'$

---

additional properties

So, within the upper part there occur only two identifiers! It remains to give the additional properties. Actually, there is only one, namely that the extensions stay the same:

$$E1\_exn = E1\_exn';\ \ \ldots;\ \ E6\_exn = E6\_exn';\ \ R1\_exn = R1\_exn';\ \ R2\_exn = R2\_exn'$$

By the way, the sets *Id* and *Val* have been postulated, and are therefore fixed throughout this discussion, and similarly for the pairing function *pair*.

Special state changes. For one reason or another, we might come across the need to characterize special state changes, that is, state changes that at least satisfy the properties of $\Delta State$ (thus being valid changes) and moreover satisfy another property of interest. For example, consider the property that relation $R1$ only "gets larger". These state changes are characterized by the following schema:

Growing_State

$\Delta State$

$R1\_ext \subseteq R1\_ext'$

The informal suggestion is ambiguous; a different formalization of the extra property is:

$$\#R1\_ext \subseteq \#R1\_ext'$$

The modular approach

Since our goal was to show how, mathematically, a formal semantics of ERDs would look like, and not to present the beauty of the Z notation, we will be very brief here.

In the modular approach we define schemas for different aspects and then combine them, mainly by logical conjunction, into the desired schema *State*. In this way, *State* is a schema *expression* in the Z notation that is *equivalent* to schema *State* given earlier.

Here are some examples of the little schemas that we might define:

For each entity and relation a schema to formalize the type
(the attributes and their possible values).
For each entity and relation a schema to formalize the extension.
For each entity and relation a schema to formalize the extent and attribute functions.
For each multiplicity property a schema.
For each static and dynamic specialization a schema.
For each comment attached to the ERD a schema.

Some of these refer, in the upper part, to others, like our schema $\Delta State$ has *State* and $State'$ in its upper part.

Furthermore, there are some formulas or formula patterns that will occur over and over again, when formalizing various ERDs. In the Z notation it is possible to write one *generic* schema for such a formula pattern, then put that into a library, and *use* it over and over again by just referring to it rather than duplicating it.

Conclusion

Although the principal interpretation of ERDs is a property of the real world and thus *informal*, the ERD notation does have a formal semantics of its own. Such a formal semantics may help to come to better understanding of some aspects of ERDs, and is almost a necessity when building tools that manipulate ERDs or assist in such manipulations.

If manipulation of the formal semantics is an additional aim, which will rarely be the case, then the Z notation is a good choice since it facilitates compact formulation and comes with a set of tools (type-checkers, theorem provers) that may be beneficial.

References

[1] J.M. Spivey. *The Z notation: a reference manual (2nd edition).* Prentice Hall International, UK, 1992.

[2] R.J. Wieringa. *Design Methods for Reactive Systems — Yourdon, Statemate, and the UML.* University of Twente, Enschede, Netherlands, 2001. To appear.