Documents: A Domain Analysis[1]

An Experiment in Domain Engineering

Version 3

Dines Bjørner

Computer Science and Engineering
Informatics and Mathematical Modelling
Technical University of Denmark
DK-28000 Kgs.Lyngby
Denmark

Graduate School of Information Science
Japan Adv. Inst. of Science & Technology
1-1, Asahidai, Tatsunokuchi
Nomi, Ishikawa 923-1292
Japan

bjorner@gmail.com

July 27, 2006. Compiled August 7, 2006

Presentation @ IBM TRL Wednesday 9 August 2006
IBM Japan Ltd., 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken, 242-8502 Japan

[1] Paper to be presented at IBM TRL, Wedn. 9 August 2006

0. Abstract

- We aim at exemplifying applications of Domain Engineering.
- That is, to describe, informally and formally, typically, a man-made universe of discourse.
- As it is. Without any reference
  - ⋆ to requirements to any computing system,
  - ⋆ let alone software to support activities in that domain.
- The domain is that of documents.
- We wish to understand what documents are:
  - ⋆ original and master documents, versions, copies, etc.,
  - ⋆ document creation, editing, reading, copying, etc.
- The idea of domain modelling and the style used in this presentation is basically the same — more-or-less independent of the domain.

0. Abstract (Continued)

- The current $\boxed{\text{Version 3}}$ of this report is skimpy on
  - ⋆ the issue of system space of documents (and agenets),
  - ⋆ of the dynamic (time-wise) movement of documents, and hence
  - ⋆ on all the very many constraints that govern documents and their locations.

1. Introduction

1.1 Aims and Objectives

- The abstract said it all.
- But we can reiterate some points.
  - ⋆ In software development we proceed
    - ◇ from modelling the application or business domain,
    - ◇ via constructing requirements from the domain model,
    - ◇ to designing the computing system, including software.
  - ⋆ We naturally wish to understand the business area well before we embark on requirements.

1.1.1 Aims

- So the aims are to show you aspects of domain modelling.
- What it includes doing.
- How one might go about doing it (including analysis).

1.1.2 Objectives

- And the objectives are
  - ⋆ to convince you to do likewise.
  - ⋆ To do software development professionally.

1.2 Domain Engineering

- Domain engineering is the engineering, the managed construction of domain descriptions.
- As such domain engineering includes

   1. identifying, contacting, and throughout consulting all possibly relevant domain stakeholders;
   2. domain knowledge acquisition;
   3. analysing acquired domain knowledge;
   4. creating the domain model[2];

   5.–6. verifying and then validating the model[3];

   7. and possibly turning the domain model into a domain theory[4].

- In this talk we shall only illustrate item 4.
- We refer to Part III (Chaps. 8–16) of volume 3 of the three volume book on Software engineering by Dines Bjørner for "the 'full' story" on Domain Engineering.

[2] A domain description is a syntactic construction. A domain model is a selected meaning (semantics) of that description.
[3] Verfication shall ensure that the model is correct (i.e., right). Validation shall ensure that we have the right model.
[4] A domain theory is the domain model + a(ny) number of theorems about the model: its properties, so to speak.

1.3 Related Work

- For the kind of document domain analysis, as we delineate it, there is, to our knowledge, no related work.
- Well, there is probably some AI-related knowledge engineering (KIF[5]) work and there is definitely some Web-oriented (for example, DAML[6] Web-ontology) on 'documents'.
- But it appears that all of this work is about the textual content of documents — well-nigh the only thing we are not interested in modelling in this presentation.
- That is, our model is not about the textual (statistical, linguistical or other) properties of documents,
- but of how documents are handled irrespective of their content.

[5] http://logic.stanford.edu/kif/kif.html
[6] The DAML (The DARPA Agent Markup Language) homepage is http://www.daml.org/

1.4 A Caveat

- This is, in principle, a first complete draft of a report, perhaps eventually a publishable paper, on the issue of domain analysis of one view of the domain of documents.
- It is based on various fragments, some appearing in the authors three volumes on software engineering, others appearing in draft versions of a report worked on during the early spring of 2006 at JAIST.
- The present draft report is a complete rewrite of those earlier fragments — with no rereading of these having beein done immediately before or during the writing of the current report.
- It is released to colleagues and some IBM TRL staff before or on 9 August 2006.
- It is much expected that the author will have time to polish this report into a possibly publishable paper during the early fall of 2006.

1.5 Structure of Talk

- There are basically three sub-parts to the main part of this report.
  - ⋆ In the first we analyse basic aspects of documents: whick kinds of dicuments there are, the operations on documents, document history, etc.
  - ⋆ In the second part we analyse notions of locations, time and agents.
  - ⋆ And in the third part we enlarge the scope to that of spaces of documents and agents.
- A concluding sub-part reviews what has been done, the possible shortcomings of the present work, and hopes for future improvement and extensions.

2. Documents: A Domain Analysis

2.1 Basics

2.1.1 Analysis: Basic Entities

- Think of documents as stapled collections a sheets of paper.
- From blank sheets you can create a basically blank document.
- You and others can write something on the sheets (i.e., edit them).
- You and others can read these sheets, i.e., the document.
- You can copy a document. Lets call the document being copied the master.
  - ⋆ Now two documents exists, one per copying action.
  - ⋆ To make n (say 100) copies is to copy the either the master a n times, or to mix copying copies (which thereby become "new masters"), or the "original" master.
  - ⋆ The point is: You know what you are doing: which way you did the copying.
- Our domain model of documents must capture that.

2.1.1 Analysis: Basic Entities (Continued)

- You can edit a document.
  - ⋆ After editing the document upon which the editing was done no longer “exists”.
  - ⋆ An edited version has been made of that “prior” document.
  - ⋆ If, say, that basis document (the prior one) was typed, and your editing was made by pencil,
    - ◇ then from the edited version you can, in a sense, “capture” (“just now edited”) un-edited (the base, prior) document.
    - ◇ In a sense you can “undo” the editing.
- Our domain model of documents must capture that.

2.1.1 Analysis: Basic Entities (Continued)

- You can read a document, say at location ℓ and time τ.
  - ⋆ Before you read it (at location ℓ and time τ) the document was known to have not been read by you at location ℓ and time τ.
  - ⋆ Afterwards, that is, after time τ, it is known to have been read by you at location ℓ and time τ.
- You can move, say distribute, a document.
  - ⋆ Before it was moved it was a some location ℓ. After the move it is at some other location ℓ'.
  - ⋆ No two documents can at any time at a same location.
  - ⋆ A document cannot be at two distinct locations at any (one) time.
- Our domain model of documents must capture that.
- Our domain model of documents must capture a lot more!

2.1.2 Narrative: Basic Entities

1. We postulate sorts, i.e., a types of documents, locations, time, agents and tex.
2. With a document we can associate, i.e., we can observe
   (a) the current location of the document,
   (b) the time it was most recently operated upon,
   (c) some identity of the agent (f.ex., person) who performed the most recent opration on the document,
   (d) which kind of agent operation was last applied to the document,
   (e) and we can observe some text, "the document contents".[7]
3. The 'kinds of operations' are tokens.(See items 5 below.)

[7]Other than being able to observe the text we shall not deal any further with format, statistical or linguistic properties of texts.

2.1.3 Formalisation: Basic Entities

type

1. D, L, T, A

value

2a. obs_L: D $\rightarrow$ L

2b. obs_T: D $\rightarrow$ T

2c. obs_A: D $\rightarrow$ A

2c. obs_OpKind: D $\rightarrow$ OpKind

2e. obs_Txt: D $\rightarrow$ Txt

type

3. OpKind == cr|ed|rd|cp|sh|mv|...

2.1.4 Narrative: Basic Functions

4. From a document one can observe the triple of location, time and performing agent of ‘most recent operation’.
5. Documents can be
   (a) created,
   (b) edited,
   (c) read,
   (d) copied,
   (e) shredded,
   (f) moved,
   (g) and a few other things (...).[8]

[8] We shall postpone till later considerations of moving (distributing), sharing , calculating over (incl. searching for) and tracing documents.

2.1.4 Narrative: Basic Functions (Continued)

6. Editing a document d results in a pair, $(d', (f_e, b_e))$:
   (a) f_e is a forward editing function.
   (b) b_e is the inverse, "backward" editing function,
   (c) and $d' = f_e(d)$ and $d = b_e(d')$,
   (d) that is, $f_e(b_e(d')) = d'$ and $b_e(f_e(d)) = d$.
   (e) The pair (f_e, b_e) is a result of the editing. It can only be known a posteriori.

2.1.5 Formalisation: Basic Functions

type

L, T, A, E

$\mathrm{LTA} = \mathrm{L} \times \mathrm{T} \times \mathrm{A}$

value

5a. crea: $\mathrm{LTA} \to \mathrm{D}$

5b. edit: $\mathrm{LTA} \to \mathrm{D} \to \mathrm{D} \times \mathrm{E}$

5c. read: $\mathrm{LTA} \to \mathrm{D} \to \mathrm{D}$

5d. copy: $\mathrm{LTA} \to \mathrm{D} \to \mathrm{D} \times \mathrm{D}$

5e. shrd: $\mathrm{LTA} \to \mathrm{D} \to$ **Unit**

4. obs_LTA: $\mathrm{D} \to \mathrm{LTA}$

2.1.6 Formalisation: The Editing Functions

type

$E = FE \times BE$

6a. $FE = D \to D$

6b. $BE = D \to D$

value

7. $\mathrm{edit}\colon LTA \to D \to D \times E$

axiom

$\forall\, d{:}D,(fe,be){:}E \cdot d = be(fe(d))$, i.e.,: $fe^\circ be{=}\lambda d.d \wedge be^\circ fe{=}\lambda d.d$

i.e.:

$\forall\, lta{:}LTA,d{:}D \cdot$ **let** $(d',(fe,be)) = \mathrm{edit}(lta)(d)$ **in**

6c. $d' = fe(d) \wedge d = be(d')$ **end**

i.e.:

6d. $d' = fe(be(d')) \wedge d = be(fe(d))$

2.1.7 Analysis: Document Identifiers

- Let's do a bit more analysis.
- Documents which have just been created are referred to as originals.
- With any original document we can associate a unique document identifier.
  - ⋆ You may think of the unique document identifier being a simple isomorphic function of three quantities:
    - ◇ the identity of the agent who (or which) created the document,
    - ◇ a unique representation of the location at which the document was created, and
    - ◇ a unique representation of the time at which the document was created.
- No agent can be at two locations at any one time and perform more than one operation at a time.and operations requires agents, hence all original document identifiers are unique.
- We shall soon see that no two otherwise distinct documents which may exist at any time can have the same unique identifier.

2.1.7 Analysis: Document Identifiers (Continued)

- Thus there are orginal ("blank") documents distinguishable, albeit, by unique document identifiers.
- Editing a document need not change its identification. Its is, for example, still "one and the same" sheet of papers, albeit with, perhaps, a different textual contents[9].
- Reading and moving a document need not change its identification either.
- Copying a document results in one more document:
  - ⋆ the master whose identification could (hence in our model will) be the same, since it is "physically" the same, as the document being copied, and
  - ⋆ the copy, the, or a, "new" document — whose identification must be made unique, hence different from the identification of the master document.
  - ⋆ We suggest that the identification of the copy document be an isomorphic function of the identification of the master document and the triplet identification of the location, the location and the agent of copying.
- Since no agent can be performing more than one operation at a time all documents will have unique identifiers.

[9] And, if not a collection of sheets of paper, then it might be an electronic document for which we would want the same property as for the humanly manifest document. Yes? Yes!

2.1.8 Formalisation: Document Identifiers

type
  L, T, A, D, E, UDI
  $\mathrm{LTA} = \mathrm{L} \times \mathrm{T} \times \mathrm{A}$
value
  crea: $\mathrm{LTA} \rightarrow$ **Unit** $\rightarrow \mathrm{D}$
  edit: $\mathrm{LTA} \rightarrow \mathrm{D} \rightarrow \mathrm{D} \times \mathrm{E}$
  read: $\mathrm{LTA} \rightarrow \mathrm{D} \rightarrow \mathrm{D}$
  copy: $\mathrm{LTA} \rightarrow \mathrm{D} \rightarrow \mathrm{D} \times \mathrm{D}$
  ...
  obs_UDI: $\mathrm{D} \rightarrow \mathrm{UDI}$
  udi: $(\mathrm{LTA}|(\mathrm{LTA}\times\mathrm{LTA})) \rightarrow \mathrm{UDI}$
axiom
  $\forall\ \ell$ta:LTA,d:D,e:E •
    obs_UDI(crea(ℓta))=udi(ℓta) $\wedge$
    obs_UDI(d) = **let** (d′,efs)=edit(ℓta)(d) **in** obs_UDI(d′) **end** $\wedge$
    obs_UDI(d) = obs_UDI(read(ℓta)(d)) $\wedge$
    let (md,cd) = copy(ℓta)(d) **in**
      obs_UDI(d) = obs_UDI(md) $\wedge$
      obs_UDI(cd) = udi(obs_UDI(md),ℓta) $\wedge$ ... **end**

The last line of the above axiom does not guarantee uniqeness across all documents. That issue will be addressed below.

2.1.9 Analysis: Document Kinds

- So there are originals, that is, documents that have been created, but upon which no actions have so far been performed.
- Then there are documents which have been edited, read, are copies of masters, and hence there are master documents; and there are documents which have been moved.
- For the time being we shall be content with just these kinds of documents (and hence 'kind of operation' tokens).
  - ⋆ An edited document may be textually distinct from the document from which it was edited.
  - ⋆ A document which has been read (at some location and time and by some agent) is, textually, the same as when that document had not been read (at that location and time and by that agent).
  - ⋆ A document which has been copied becomes a master document, otherwise textually indistinguishable from the document from which the copy was made.
  - ⋆ The copy of a master document is a copied document and is distinguishable, not textually, but "kind-wise", from the master document.

2.1.10 Analysis: Document History

- So "most recently" edited documents are distinct textually distinct from the documents from which they were edited.
- The reason, that is, the pragmatics behind why we introduce the notion of 'document kind' is so that we can reason about documents:
  - ⋆ "That document has been read, by such and such, several times."
  - ⋆ "That document is an edited version of a document which we was read by agent α, from a document which was copied by ..., etc."
- We wish to also be able to also reason about
  - ⋆ locations and
  - ⋆ times

  of most recent actions, and about
  - ⋆ the document as it was before such an action:
  - ⋆ "the document from which it resulted".

2.1.10 Analysis: Document History (Continued)

- Although in today's practical life one may not record, on documents,
  - ⋆ who first created them (where and when),
  - ⋆ who edited them (where and when),
  - ⋆ who read them (where and when),
  - ⋆ who copied them (where and when),
  - ⋆ or from which masters they were copied (where and when),
  - ⋆ etc.,

  we may still, in this everyday world, be able to reason about such things.
- Therefore we must model it.[10]

[10] The issue of the certainty (i.e., uncenrtainty) with which we may say reason will be discussed later.

2.1.11 Narrative: Document Kinds

7. Thus documents are "most recently" either
   (a) created, i.e., they are [blank] "originals",
   (b) edited,
   (c) read,
   (d) the basis for copying, i.e., they are "masters", or
   (e) the result of copying, i.e., they are copies,
   (f) etcetera.

2.1.12 Formalisation: Document Kinds

type

7 mD,D = oD | eD | rD | cD

2.1.12 Formalisation: Document Kinds (Continued)

axiom

$\forall$ odoc:oD,edoc:eD,rdoc:rD,cdoc:cD •

7a obs_OpKind(odoc) = cr

7b obs_OpKind(edoc) = ed

7c obs_OpKind(rdoc) = rd

7d obs_OpKind(cdoc) = cp

$\forall$ lta:LTA,d:D •

7a obs_OpKind(crea(lta)) = cr

7b obs_OpKind(edit(lta)(d)) = (ed,efs)

7c obs_OpKind(read(lta)(d)) = rd

7d **let** (d′,d″)=copy(lta)(d) **in** obs_OpKind(d′)=cp$\wedge$obs_OpKind(d′)$\in${cr,ed,rd,cp,...} **end**

...

2.1.13 Narrative: Document History

8. With a document we can associate its history of operations performed on that document. More specifically we can think of a document history to be a list of triples: the operation performed, the document, and a location, time and agent triple.
9. An original document has no pre-history. It has the history that it was created at such and such a location and time by such and such an agent.
10. An edited document has a history which consists of the its pre-history, namely the history of the document from which it was edited, and then the most recent historical fact, namely that it was edited at such and such a location and time by such and such an agent, and that the editing can be captured, i.e., the text of the document from which it was edited, as well as the text resulting from the editing.
11. A document which has "just" been read has the pre-history of the document before it was "just" read as well as the most recent historical fact (location, time and agent of reading).

2.1.13 Narrative: Document History (Continued)

12. The two documents which have partaken in a copying action: the master and the copy documents have different "most recent" histories, but the same pre-history:
    (a) The master copy has, as "most recent" history that it served as a master for a copying action (by whom, when and where).
    (b) The copy has, as "most recent" history that it was the result of a copying action (by whom, when and where).
    (c) Since the unique identifier of the copy "embeds" that of the master one can identify the master.
    (d) Perhaps one should "insert" as part of the "most recent" master history the unique identifier of the copy: that identifier can anyway be constructed!
13. Etcetera.

2.1.14 Formalisation: Document History

type

8 $\mathrm{H} == \mathrm{D}^*$

value

8 $\mathrm{obs\_H}\colon \mathrm{D} \to \mathrm{H}$

$\mathrm{wf\_H(dl)} \equiv$ **len** $\mathrm{dl}\geq 1 \wedge \mathrm{dl}(1)=\mathrm{create(lta)} \wedge \mathrm{create(lta)} \notin$ **elems tl** dl

axiom

$\forall$ lta:LTA,d:D $\cdot$

9 $\mathrm{obs\_H(create(lta))} = \langle \mathrm{create(lta)} \rangle \wedge$

10 $\mathrm{obs\_H(edit(lta)(d))} = \mathrm{obs\_H(d)} \frown \langle \mathrm{edit(lta)(d)} \rangle \wedge$

11 $\mathrm{obs\_H(read(lta)(d))} = \mathrm{obs\_H(d)} \frown \langle \mathrm{read(lta)(d)} \rangle \wedge$

12 **let** (md,cd) = copy(lta)(d) **in**

12a $\mathrm{obs\_H(md)}=\mathrm{obs\_H(d)} \frown \langle \mathrm{md} \rangle \wedge$

12b $\mathrm{obs\_H(cd)}=\mathrm{obs\_H(d)} \frown \langle \mathrm{cd} \rangle \wedge$

12c ... **end**

2.1.14 Formalisation: Document History (Continued)

- The axiom above essentially defines an observer function

value
  obs_prev_D: D → D | nil
axiom
  obs_prev_D(create(lta)) = nil ∧
  obs_prev_D(edit(lta)(d)) = d ∧
  obs_prev_D(read(lta)(d)) = d ∧
  let (md,cd)=copy(lta)(d) **in** obs_prev_D(md)=d ∧ obs_prev_D(cd)=d **end** ∧
  ...

- In other words: It is, of course, always possible, in the domain,
- to reason about past versions versions of a document.
- Whether one's reasoning is accurate — one might have forgotten essential aspects — is another matter[11]

[11]— which we should model by introducing non-determinism into our description, but we will leave that to a full, formal paper!

2.2 Locations, Time and Agents

- We have mentioned that we do not study textual issues.
- But we are concerned about locations, time and agents.
- There are two notions of location:
  - ⋆ the location at which an operation on a document took, or takes place,
  - ⋆ and the location of the document "right now".
  - ⋆ The latter is, of course (?), the same as the location to where the most recent move action brought that document!
- There are two similar notions of time
  - ⋆ the time at which an operation on a document took, or takes place,
  - ⋆ and the time "right now", or at any point in the oast, or at any point in the future.
- And there is a notion of agent: the person (or other) who (resp. which) performs actions on documents.
- A document is "full" of location, time and agent attributations.
- The location, time and agent notions deserve a special study.

2.2.1 Locations

2.2.1.1 Analysis: Points

- We assume a space as a dense set of points,
- with a point being atomic, but not necessarily infinitisimally "small" in the physical sense of XYX dimension units.
- Thus any two points that are distinct at most shares that they may be more-or-less close, or more-or-less distant from one another.

2.2.1.1 Analysis: Points (Continued)

- We may even introduce a notion of some point being "next to" another point,
  - ⋆ in which case there will probably be an indefinite if not infinite number of points close to a given point, and such that
    - ◇ if two points
      - ∘ π_i and π_j are next to one another
      - ∘ and π_j and π_k are also next to one another,
    - ◇ then
      - ∘ either π_i is the same as π_k, or π_j is the same as π_k,
      - ∘ or they are all three distinct.

2.2.1.2 Narration: Locations

14. By a location we shall understand a dense set of points,
    (a) such that for every two distinct points π_i and π_k in the location (if the location contains more than one point)
        i. there is an ordered set of at least two distinct points in the location, $\{\pi_i, \pi_{i_1}, \ldots, \pi_{i_j}, \ldots, \pi_{i_{k-1}}, \pi_{i_k}\}$,
        ii. such that each pair of these points, $\pi_{i_{j-1}}, \pi_{i_j}$, or $\pi_{i_j}, \pi_{i_{j+1}}$,
        iii. for $i \leq j-1$ and $j \leq k$,
        iv. are 'next to' one another.

2.2.1.2 Narration: Locations (Continued)

15. Two locations are ‘different’ if the two sets are different.
16. Two locations are ‘distinct’ if they are different and they have no points in common (that is, share no points).
17. Two locations are ‘next to’ one another
    (a) if they are different,
    (b) if there is at least a non-empty set of points in each location, ps_i and ps_j
    (c) whose points are “not in the location of the other”,
    (d) such that pairs of points in ps_i and ps_j are “next to” one another
    (e) and such that all other points in the two locations are shared.
18. Two locations are ‘neigbouring’ if they are distinct and there is at least one point in each location, p_i and p_j such that these are “next to” one another.

2.2.1.3 Formalisation: Locations

type

14 P, L

value

14 obs_Ps: L $\rightarrow$ P-**set**

axiom

14 $\forall$ ℓ:L $\cdot$ **let** ps = obs_Ps(ℓ) **in**

14a $\forall$ pi,pk:P $\cdot$ {pi,pk}$\subseteq$ps $\cdot$

14(a)i **let** pl:P* $\cdot$ **elems** pl = ps $\wedge$ **len** pl=**card** ps **in**

14(a)ii $\forall$ j:**Nat** $\cdot$ {j,j+1}$\subseteq$**inds** pl

14(a)iii next_to(pl(j),pl(j+1))

14(a)iv **end end**

2.2.1.3 Formalisation: Locations (Continued)

value

15 $\text{diff}(\ell_i,\ell_j) \equiv \text{obs\_Ps}(\ell_i) \neq \text{obs\_Ps}(\ell_j)$

16 $\text{dist}(\ell_i,\ell_j) \equiv \text{obs\_Ps}(\ell_i)\text{isect obs\_Ps}(\ell_j)=\{\}$

17 $\text{next\_to}(\ell_i,\ell_j) \equiv$
17a $\quad \text{diff}(\ell_i,\ell_j)\wedge$
17b $\quad \exists\ \text{ps\_i,ps\_j:P-}\textbf{set}\cdot\text{ps\_i}\subseteq\text{obs\_Ps}(\ell_i)\wedge\text{ps\_j}\subseteq\text{obs\_Ps}(\ell_j)\wedge$
17c $\quad\quad \forall\ \text{p\_i,p\_j:P:p\_i} \in \text{ps\_i}\wedge\text{p\_j} \in \text{ps\_j}\cdot\text{p\_i}\notin \text{obs\_Ps}(\ell_j)\wedge\text{p\_j}\notin \text{obs\_Ps}(\ell_i)\wedge$
17d $\quad\quad\quad \text{next\_to(p\_i,p\_j)}\wedge$
17e $\quad\quad\quad \text{obs\_Ps}(\ell_i)\backslash\{\text{ps\_i}\}=\text{obs\_Ps}(\ell_j)\backslash\{\text{ps\_j}\}$

15 $\text{diff: L} \times \text{L} \rightarrow$ **Bool**
16 $\text{dist: L} \times \text{L} \rightarrow$ **Bool**
17 $\text{next\_to: (P}\times\text{P)} \mid (\text{L}\times\text{L}) \rightarrow$ **Bool**

2.2.1 Formalisation: Locations (Continued)

18 neighbour: L $\times$ L $\rightarrow$ **Bool**
18 neighbour(ℓ_i,ℓ_j) $\equiv$
18   distinct(ℓ_i,ℓ_j) $\wedge$
18   $\exists$ ps_i,ps_j:P-**set** $\cdot$
18   ps_i$\subseteq$obs_Ps(ℓ_i)$\wedge$ps_j$\subseteq$obs_Ps(ℓ_j)$\wedge$ps_i$\neq${}$\wedge$ps_j$\neq${}$\wedge$
18     $\forall$ p_i,p_j:P:p_i $\in$ ps_i$\wedge$p_j $\in$ ps_j$\cdot$p_i$\notin$ obs_Ps(ℓ_j)$\wedge$p_j$\notin$ obs_Ps(ℓ_i)$\wedge$
18   next_to(p_i,p_j)

2.2.2 Time and Time Intervals

2.2.2.1 Analysis: Time

- Time, besides philosophically being an elusive issue, can also, as here, be taken concretely.
- We shall present a set of axioms due to Johan van Benthem.
- We shall think of time to be a notion that is absolute wrt. some calendar, that is, time includes year, month, day, hour, minute, second, etc., in some simple encoded form.

2.2.2.2 Formalisation: Time

axiom

[TRANS: Transitivity] $\forall$ p,p′,p″:P $\cdot$ p $<$ p′ $<$ p″ $\Rightarrow$ p $<$ p″

[IRREF: Irreflexitivity] $\forall$ p:P $\cdot$ p $\not<$ p

[LIN: Linearity] $\forall$ p,p′:P $\cdot$ (p=p′ $\vee$ p<p′ $\vee$ p>p′)

[L−LIN: Left Linearity]
$\forall$ p,p′,p″:P $\cdot$ (p′<p $\wedge$ p″<p) $\Rightarrow$ (p′<p″ $\vee$ p′=p″ $\vee$ p″<p′)

[BEG: Beginning] $\exists$ p:P $\cdot$ $\sim\exists$ p′:P $\cdot$ p′<p

[END: Ending] $\exists$ p:P $\cdot$ $\sim\exists$ p′:P $\cdot$ p<p′

2.2.2.2 Formalisation: Time (Continued)

[SUCC: Successor]
  [PAST: Predecessors] $\forall$ p:P,$\exists$ p′:P $\cdot$ p′<p
  [FUTURE: Successor] $\forall$ p:P,$\exists$ p′:P $\cdot$ p<p′

[DENS: Dense] $\forall$ p,p′:P (p<p′ $\Rightarrow$ $\exists$ p″:P $\cdot$ p<p″<p′)

[DENS: Converse Dense] $\equiv$ [TRANS: Transitivity]
  $\forall$ p,p′:P ($\exists$ p″:P $\cdot$ p<p″<p′ $\Rightarrow$ p<p′)

[DISC: Discrete]
  $\forall$ p,p′:P $\cdot$ (p<p′ $\Rightarrow$ $\exists$ p″:P $\cdot$ (p<p″ $\wedge$ $\sim\exists$ p‴:P $\cdot$ (p<p‴<p″))) $\wedge$
  $\forall$ p,p′:P $\cdot$ (p<p′ $\Rightarrow$ $\exists$ p″:P $\cdot$ (p″<p′ $\wedge$ $\sim\exists$ p‴:P $\cdot$ (p″<p‴<p′)))

2.2.2.3 Comments on the Axiomatisation: Time

- A strict partial order, SPO, is a point structure satisfying TRANS and IRREF.
- TRANS, IRREF and SUCC imply infinite models.
- TRANS and SUCC may have finite, “looping time” models.
- We choose the SPO model of time.

2.2.2.4 Analysis: Time Intervals

- Time is "absolute" wrt. some "zero".
- A time interval may be measured in years, months, etc., but these are year intervals, month intervals, etc., not "absolute" wrt. some calendar.
- When we subtrate one time (a date etc.) from another such time we get a time interval.
- We cannot add time, but we can add a time interval to a time and get a time.
- We can add, subtract and multiply time intervals.
- If we divide a time interval by a non-zero such we get a fraction, i.e., a real number.
- Etcetera.
- At the stage of this paper (/today) we shall not need the notion of time interval, nor the operations on times.

2.2.2.5 Formalisation: Time and Time Intervals

type
T, TI
value
$\mathrm{elapsed\_time}\colon \mathrm{T} \times \mathrm{T} \to \mathrm{TI}$
$-\colon (\mathrm{T}|\mathrm{TI}) \times \mathrm{TI} \to \mathrm{TI}$
$-\colon \mathrm{T} \times \mathrm{T} \to \mathrm{TI}$
$+\colon \mathrm{T} \times \mathrm{TI} \to \mathrm{T}$
$+\colon \mathrm{TI} \times \mathrm{TI} \to \mathrm{TI}$
$*\colon \mathrm{TI} \times \mathbf{Real} \to \mathrm{TI}$
$/\colon \mathrm{TI} \times \mathrm{TI} \xrightarrow{\sim} \mathbf{Real}$
axiom
$\forall\ t,t'{:}\mathrm{T} \cdot t'>t \Rightarrow \exists\ t\delta{:}\mathrm{TI} \cdot t\delta = t'-t$
$\forall\ t{:}\mathrm{T},t\delta{:}\mathrm{TI} \cdot \exists\ t'{:}\mathrm{T} \cdot \Rightarrow t+t\delta = t' \vee t-t\delta = t'$
$\forall\ ti,ti'{:}\mathrm{TI} \cdot \exists\ ti''{:}\mathrm{TI} \cdot ti+ti' = ti'' \vee\ ti-ti' = ti''$
...

2.2.3 Time and Space

2.2.3.1 Analysis: Time and Space

- We show an analysis of some time/spce notions.
- The analysis is due to Wayne D. Blizard.
- In the axiomatisation next entities are the documents (or agents, A and B) and points are (our) locations (p, q and r), with
- A_q^t expresses that an entity A at time t is at location p (axiom (I)).
- Hear our comments next.

2.2.3.2 Formalisation: Time and Space

$$
\begin{array}{rrcll}
(I) & \forall A \forall t \exists p & : & A_p^t & \\
(II) & (A_p^t \wedge A_q^t) & \supset & p = q & \\
(III) & (A_p^t \wedge B_p^t) & \supset & A = B & \\
(IV) & (A_p^t \wedge A_p^{t'}) & \supset & t = t' & \\
(V\ \ i) & \forall p, q & : & N(p,q) \supset p \neq q & \text{Irreflexivity} \\
(V\ \ ii) & \forall p, q & : & N(p,q) = N(q,p) & \text{Symmetry} \\
(V\ \ iii) & \forall p \exists q, r & : & N(p,q) \wedge N(p,r) \wedge q \neq r & \text{No isolated pts.} \\
(VI\ \ i) & \forall t & : & t \neq t' & \\
(VI\ \ ii) & \forall t & : & t' \neq 0 & \\
(VI\ \ iii) & \forall t & : & t \neq 0 \supset \exists \tau : t = \tau' & \\
(VI\ \ iv) & \forall t, \tau & : & \tau' = t' \supset \tau = t & \\
(VII) & A_p^t \wedge A_q^{t'} & \supset & N(p,q) & \\
(VIII) & A_p^t \wedge B_q^t \wedge N(p,q) & \supset & \sim (A_q^{t'} \wedge B_p^{t'}) &
\end{array}
$$

2.2.3.3 Annotations: Time and Space

- (II–IV,VII, VIII): The axioms are universally 'closed', that is, we have omitted the usual $\forall A, B, p, q, t$s.
- (I): For every entity, A, and every time, t, there is a location, p, at which A is located at time t.
- (II): An entity cannot be in two locations at the same time.
- (III): Two distinct entities cannot be at the same location at the same time.
- (IV): Entities always move: An entity cannot be at the same location at different times. *This is more like a conjecture, and could be questioned.*
- (V): These three axioms define N.
- (V i): Same as $\forall p :\sim N(p,p)$. "Being a neighbour of", is the same as "being distinct from".
- (V ii): If p is a neighbour of q, then q is a neighbour of p.
- (V iii): Every location has at least two distinct neighbours.

2.2.3.3 Annotations: Time and Space (Continued)

- (VI): The next four axioms determine the time successor function $'$.
- (VI i): A time is always distinct from its successor: Time cannot rest. There are no time fix points.
- (VI ii): Any time successor is distinct from the begin time. Time 0 has no predecessor.
- (VI iii): Every nonbegin time has an immediate predecessor.
- (VI iv): The time successor function $'$ is a one-to-one (i.e., a bijection) function.
- (VII): The *continuous path axiom:* If entity A is at location p at time t, and it is at location q in the immediate next time t', then p and q are neighbours.
- (VIII): *No "switching":* If entities A and B occupy neighbouring locations at time t the it is not possible for A and B to have switched locations at the next time t'.

2.2.3.4 Discussion of the Blizard Model of Space/Time

- Except for axiom (IV) the system applies to systems of entities that "sometimes" rest, i.e., do not move.
- These entities are spatial and occupy at least a point in space.
- If some entities "occupy more" space volume than others, then we may suitably "repair" the notion of the point space P (etc.),
- however, this is not shown here.

2.2.4 Agents

2.2.4.1 Analysis: Agents

- Agents create and perform operations on documents.
- Typically agents are humans — but could be machines, i.e., computers.

2.2.4.2 Narrative: Agents

19. An agent, besides having an identity, a:A, is a further undefined notion.
20. No two agents have the same identity, so if two arbitrarily chosen agents (one could "choose" the same agenet twice) are different, then they ar at least different because of their distinct identity.
21. One could choose to associate various other attributes with agents:
    (a) the set of documents in possession of the agent, say referenced by unique document identifiers,
    (b) the current location of the agent,
        i. with the possibility of also constraining agents to occupy distinct locations,
    (c) the granted authority to perform certain (or all) operations on certain (or all), including creating, documents

    but we choose to not deal with that issue in this presentation.

2.2.4.3 Formalisation: Agents

type

19. AG, A

value

19. obs_A: AG → A

axiom

21. $\forall$ a,a′:A $\cdot$ a=a′ $\equiv$ obs_A(a)=obs_A(a′) $\wedge$ obs_A(a)$\neq$obs_A(a′)$\Rightarrow$a$\neq$a′

value

21a. obs_Ds: AG → D-**set**

21b. obs_L: AG → L

axiom

21(b)i. $\forall$ a,a′ $\cdot$ a$\neq$a′ $\Rightarrow$ distinct(obs_L(a),obs_L(a′))

type

Auth

value

21c. obs_Auths: AG → Auth-**set**

2.3 Spaces of Documents and Agents

2.3.1 Narrative: Spaces of Documents and Agents

22. There is a concept of space.
23. Documents and agents coexist in space.
24. All documents and all agents in that space
    (a) have distinct unique document identifiers and
    (b) distinct agent identifiers.
25. Some properties of documents can now be re-expressed

    No two document can have the same
    i. "most recent time and location",
    ii. no two document can have the same "most recent location and agent", and
    iii. no two document can have the same "most recent time and agent"[12].

    (a) We also need to redefine the signature of all document operations — and the related axioms.

[12] Since our axioms implicitly enjoy the universal time quantitifcation this means that no two past operations can have the take place at the same location, at the same location and time and agent.

2.3.2 Formalisation: Spaces of Documents and Agents

type

22. Ω

$LT = L \times T,\ LA = L \times A,\ TA = T \times A$

value

22. $obs\_Ds: \Omega \rightarrow D$-**set**, $obs\_As: \Omega \rightarrow AF$-**set**

$obs\_LT: D \rightarrow LT,\ obs\_LA: D \rightarrow LA,\ obs\_TA: D \rightarrow TA,$

axiom

24. $\forall\ \omega:\Omega, d, d':D,\ ag, ag':AG \cdot \{d,d'\} \subseteq obs\_Ds(\omega) \wedge \{ag,ag'\} \subseteq obs\_AGs(\omega) \Rightarrow$

24a. $d = d' \equiv obs\_UDI(d) = obs\_UDI(d') \wedge$

24b. $ag = ag' \equiv obs\_A(ag) = obs\_A(ag') \wedge$

25. $\forall\ \omega:\Omega, d, d':D \cdot \{d,d'\} \subseteq obs\_Ds(\omega) \Rightarrow$

25()i. $d \neq d' \Rightarrow obs\_LT(d) \neq obs\_LT(d') \wedge$

25()ii. $d \neq d' \Rightarrow obs\_LA(d) \neq obs\_LA(d') \wedge$

25()iii. $d \neq d' \Rightarrow obs\_TA(d) \neq obs\_TA(d') \wedge$

2.3.3 Formalisation: Types of Document Operations

value

5a. crea: $\mathrm{LTA} \to \Omega \times \mathrm{UDI}$

5b. edit: $\mathrm{LTA} \to \mathrm{UDI} \to \Omega \to \Omega \times \mathrm{E}$

5c. read: $\mathrm{LTA} \to \mathrm{UDI} \to \Omega \to \Omega$

5d. copy: $\mathrm{LTA} \to \mathrm{UDI} \to \Omega \to \Omega \times \mathrm{UDI}$

5e. shrd: $\mathrm{LTA} \to \mathrm{UDI} \to \Omega \to \Omega$

axiom

... /∗ left as an exersize ! ∗/

2.4 Dynamic System of Documents

- The dynamics of a agents and documents views these are evolving over time.
- That is, the system is a total function from time to space of agents and documents.

type
  $\mathrm{DYN} = \mathrm{T} \to \Omega$
axiom
  ...

- Axioms over dyn:DYN expresses much the same as did the axioms over space hinted at earlier.

2.5 Document Traces

- Usually creation and editing of documents are based on a possibly empty set of other documents.
- Once created or everytime edited one woild like to be able to trace, not only, as before, the document history of the created or edited document, but also those of the refeence documents.
- So a document trace is a set of document histories.
- Together the set forms a tree of histories:
  - ⋆ At the root is the document being traced.
  - ⋆ Branch nodes designate document versions, with one branch for history of each document reference.
- See Fig. 1 on the following page.

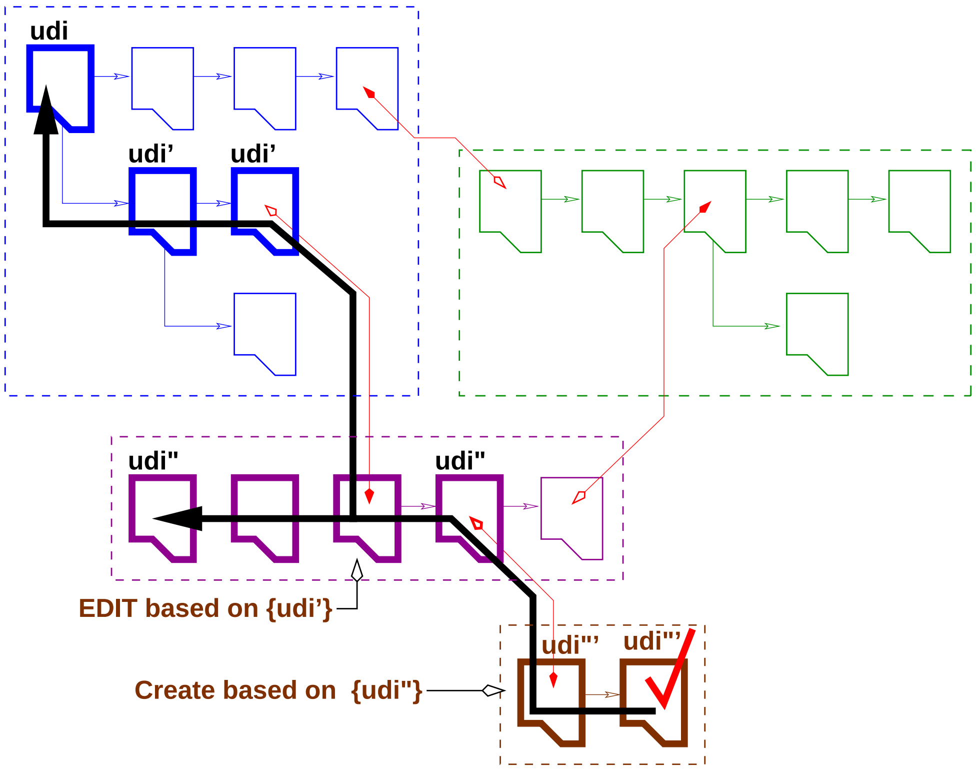

Figure 1: Document udi''' trace

- Figure 1 clearly shows a number of documents each with their operations.

2.5.1 First: Extension of Create and Edit Operations

- We extend our create and edit operations.
- When creating or editing documents we now do so on the basis of zero, one or more references to "background" documents.
- We need define an observer function, obs_UDIs_LTAs, which which when applied to any document, d yields the set of pairs of unique document identifiers (UDIs) and the location, time and agent identifier triples (LTAs) of the documents referenced in d.
- One needs the LTA triples as the document identified by the UDI may have undergone (several) operations since (first) referenced.
- Think of the LTA triples being "inserted" in the created or edited document together with the UDI —
- where these LTA elements are those of the version of the UFI identified document at the time of the create, respectively the edit operation.

2.5.1 First: Extension of Create and Edit Operations (Continued)

value
  crea: LTA × UDI-**set** → Ω → Ω × UDI
  obs_D: UDI → Ω → D
  obs_UDIs_LTAs: D → (UDI×LTA)-**set**
axiom
  ∀ lta:LTA,udis:UDI-**set** ·
    let (ω′,udi) = crea(lta,udis)(ω) **in**
    let od = obs_D(udi)(ω′) **in**
    obs_UDI(od)=udi ∧ obs_LTA(od)=lta ∧
    let udis_ltas = obs_UDIs_LTAs(od) **in**
    udis={udi′|(udi′,lta′):UDI×LTA·(udi′,lta′)∈ udis_ltas}
    end end end

Similar for editing.

2.5.2 Then: Definition of Traces

- We leave this as an exercise!

2.6 Summary

- We could go on and define additional domain notions.
  - ⋆ The notion of document family used, but not defined above.
  - ⋆ We could define concepts of document classes,
  - ⋆ document access authority,
  - ⋆ agent authorisations: by class, operation and/or document,
  - ⋆ etcetera.
- We choose to stop here.

2.6 Summary (Continued)

- In another report: *Public Government: A Domain Analysis* you will find all of the above plus more, put in the pragmatic context of the many agencies of the three branches of government:
  - ⋆ the law making (parliament, provincial and city councils),
  - ⋆ the law enforcing (state and local administration) and
  - ⋆ the law interpreting (the judiciary) branches.

3. From Domain Models to Requirements

- One rôle for $\mathcal{D}$omain descriptions is to serve as a basis for constructing , $\mathcal{R}$equirements prescriptions.
- The purpose of constructing $\mathcal{R}$equirements prescriptions is to specify properties (not implementation) of a $\mathcal{M}$achine.
  - ⋆ The $\mathcal{M}$achine is
    - ◇ the hardware (equiment) and
    - ◇ the software
  - ⋆ that together implements the requirements.
- The implementation relations is

$$\mathcal{D}, \mathcal{M} \models \mathcal{R}$$

- The $\mathcal{M}$achine is proven to implement the $\mathcal{R}$equirements in the context of [assumptions about] the $\mathcal{D}$omain.
  - ⋆ That is, proofs of correctness of the $\mathcal{M}$achine wrt. the $\mathcal{R}$equirements often refer to properties of the $\mathcal{D}$omain.

3.1 Domain Requirements

- First, in a concrete sense, you copy the domain descriptiona and call it a requirements prescriptions.
- Then that requirements prescription is subjected to a number of operations:
  - ⋆ (i) removal (projection away) of all those aspects not needed in the requiriments;
  - ⋆ (ii) instantiation of remain aspects to the specifics of the client's domain;
  - ⋆ (iii) making determinate what is unnessarily or undesirably non-deterministic in the evolving requirements prescription;
  - ⋆ (iv) extending ot with concepts not feasible in the domain; and
  - ⋆ (v) fitting these requirements to those of related domains (say monitoring & control of public administration procedures).
- The result is called a domain requirements.

3.2 Interface Requirements

- From the domain requirements one then constructs the interface requirements:
  - ⋆ First one identifies all phenomena and concepts, entities, functions, event and behaviours shared with the environment of the machine (hardware + software) being requirements specified.
  - ⋆ Then one requirements prescribe how each shared phenomenon and concept is being inistialised and updated:
    - ◇ entity initialisation and refreshment,
    - ◇ function initialisation and refreshment (interactive monitoring and control of computations), and
    - ◇ the physiological man-machine and machine-machine implements.

3.3 Machine Requirements

- Finally one deals with machine requirements
  - ⋆ performance, dependability, maintainability, portability, etc.,
  - ⋆ where dependability addresses such issues as
    - ◇ availability,
    - ◇ accessability,
    - ◇ reliability,
    - ◇ safety,
    - ◇ security,

    etc.

4. Why Domain Engineering?

4.1 Two Reasons for Domain Theories

- We believe that one can identify two almost diametrically opposed reasons for the pursuit of domain theories.
  - ⋆ One is utilitarian, concrete, commercial and engineering goal-oriented.
    - ◇ It claims that domain engineering will lead to better software,
    - ◇ and to development processes that can be better monitored and colled.
  - ⋆ and the other is science oriented.
    - ◇ It claims that establishing domain theories is a necessity,
    - ◇ that it miust be done, whither we develop software or not.
- We basically take
  - ⋆ the latter, the science, view,
  - ⋆ while, of course, noting the former, the engineering consequences.
- We will briefly look at these.

4.2 A Utilitarian, an Engineering Reason

- In a recent e-mail, in response, undoubtedly to my steadfast, perhaps conceived as stubborn insistance, on domain engineering (DE),
- Tony Hoare, possibly in order to come to grips with this "animal" (DE),
- summed up his reaction to DE as follows, and I quote[13]:

"There are many unique contributions that can be made by domain modelling.

1. The models describe all aspects of the real world that are relevant for any good software design in the area. They describe possible places to define the system boundary for any particular project.
2. They make explicit the preconditions about the real world that have to be made in any embedded software design, especially one that is going to be formally proved.

[13] E-Mail to Dines Bjørner, CC to Robin Milner et al., July 19, 2006

3. They describe the whole range of possible designs for the software, and the whole range of technologies available for its realisation.
4. They provide a framework for a full analysis of requirements, which is wholly independent of the technology of implementation.
5. They enumerate and analyse the decisions that must be taken earlier or later any design project, and identify those that are independent and those that conflict. Late discovery of feature interactions can be avoided."

- All of these issues are dealt with, in depth,
- in Vol. 3 my three volume book:
  - ⋆ *Software Engineering, Vol. 3: Domains, Requirements and Software Design.* Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.

4.3 A Scientific Reason

- But, inasmuch as the above-listed issues,
  - ⋆ so aptly expressed in Tony's mastery, also of words,

  are of course of utmost engineering importance,
- it is really, in our mind, the science issues that are foremost:
  - ⋆ We must first and foremost understand.
  - ⋆ There is no excuse for not trying to first understand
    - ◇ whether that undertanding can be "translated" into
    - ◇ engineering tools and techniques
    - ◇ is then less important.
  - ⋆ But then, of course, it is nice that understanding also leads to better engineering.
- It usually does.

5. Conclusion

5.1 What Have We Shown?

- We have established a framework for reasoning about a concept of documents:
  - ⋆ orginals,
  - ⋆ un-edited and edited documents,
  - ⋆ documents that have been read or not read,
  - ⋆ copies of master documents,
  - ⋆ etc.
- We have endowed documents with such attributes as
  - ⋆ unique document identifiers,
  - ⋆ the location, time and agent of operations performed on documents,
  - ⋆ the 'kind of operation' ("most recently") performed on documents,
  - ⋆ document history,
  - ⋆ etc.

5.1 What Have We Shown? (Continued)

- And we have "embedded" documents and agents in a space of such
  - ⋆ while expressing some, but (we think) far from all relevant properties (axiomatically)

.

- In other words, we have started establishing a domain theory for the kind of documents that are of the kinds and have the operations and properties expressed in this presentation.
- The formalisation of the domain description has been carried out using the RAISE specification labuage, RSL.
  - ⋆ Emphasis has been on an algebraic style specification
  - ⋆ using sorts, observer functions and axioms.

5.2 Shortcomings

- Immediate shortcomings are believed to be:
  - ⋆ uncertainty as to whether all relevant properties (axioms) have been formulated,
  - ⋆ uncertainty as to whether the axioms that have been expressed are consistent (they are not thought of as being complete).
- Less immediate shortcomings, we think, have to do with the following:
  - ⋆ Have we expressed the sorts, function signaturs and the axioms as succinctly as we would desire?
  - ⋆ A "gut feeling" about issues that we ought to have also covered!

5.3 What Needs Be Done?

- Obviously we need to now review the paper behind the presentations given over this subject.
- And to improve it along the lines outlined on the previous slide.
- This will take time.
- Hopefully colleagues will wish to study this the paper behind this presentation.
- And hopefully the author — and the paper — can then benefit from resulting discussions.

6. Acknowledgements

- The current author acknowledges
  - ⋆ the support of JAIST, the Graduate School of Information Science — where the preparation for the current presentation was carried out;
  - ⋆ the very preliminary discussions the author has had with colleagues, in particular Kokichi Futatsugui and Kazuhiro Ogata;
  - ⋆ and the IBM TRL for providing a stimulus for getting this presentation ready for 9 August 2006.

Contents