A Family of License Languages

Incomplete R&D Development Notes[1], June 10, 2006, 17:57

Arimoto Yasuhito[2], Dines Bjørner[1,2], Chen Xiaoyi[2], Futatsugi Kokichi[2]
Ogata Kazuhiro[2], René Vestergaard[2], Xiang Jianwen[2]

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

2: Graduate School of Information Science
Japan Advanced Institute of Science & Technology
1-1, Asahidai, Tatsunokuchi
Nomi, Ishikawa 923-1292
Japan

{arimotoy, xychen, kokichi, ogata, vester, jxiang,}@jaist.ac.jp, bjorner@gmail.com

June 14, 2006. Compiled June 10, 2006

[1]This is a "working report"-style collection of development notes, not a paper. As a "working report" it collects thoughts, ideas, experiments, formal models, observationsm, etc. This "working report" may be the basis for a JAIST reports and possibly for publishable papers. The report summarises ongoing work by its authors. The work of the 1st and 3rd authors is likewise expected to be part of their theses: MSc, respectively PhD.

0. Abstract

- Classical digital rights license languages applied to the
  - ⋆ electronic "downloading",
  - ⋆ payment and
  - ⋆ rendering (playing)

  of artistic works.
- In this document
  - ⋆ we generalise such languages
  - ⋆ and we extend the concept of licensing to also cover work authorisation (work commitment and promises) in
    - ◇ healthcare and in
    - ◇ public government.
  - ⋆ The digital works are now patient medical records and public government documents.

0. Abstract (Continued)

- Digital rights licensing for artistic works seeks to safeguard against piracy and to ensure proper payments for the rights to render these works.
- Healthcare and public government license languages seek to ensure
  - ⋆ transparent and professional (accurate and timely) healthcare,
  - ⋆ respectively 'good governance'.
- Proper mathematical definition of licensing languages seeks to ensure smooth and correct computerised management of licenses.
- In this document we shall motivate and exemplify three license languages,
  - ⋆ their pragmatics,
  - ⋆ syntax and
  - ⋆ informal as well as formal semantics.

1. Caveats

- This document constitutes a comprehended set of R&D development notes.
- It is not a report,
- let alone a JAIST report, and
- it is certainly not a publishable science and/or technology paper.
- This document is to serve as a basis for
  - ⋆ further work on the design, pragmatics, semantics and syntax of license languages,
  - ⋆ for upcoming work on understanding permissions and obligations,
  - ⋆ for upcoming work on studying possible understandings of license languages in terms of
    - ◇ game theory,
    - ◇ transaction costs,
    - ◇ and possibly other issues such as technological feasibilities.
- This document is grossly incomplete.

2. Introduction

License:

a right or permission
granted in accordance with law
by a competent authority
- to engage in some business or occupation,
- to do some act, or
- to engage in some transaction
which but for such license would be unlawful

Merriam Webster Online

2. Introduction (Continued)

- The concepts of licenses and licensing express relations between
  - ⋆ *actors* (licensors (the authority) and licensees),
  - ⋆ *entities* (works, patients, documents) and
  - ⋆ *functions* (on entities), and as performed by actors.
- By issuing a license to a licensee, a licensor wishes to express and enforce certain permissions and obligations:
  - ⋆ which functions
  - ⋆ on which entities
  - ⋆ the licensee is allowed (is licensed, is permitted) to perform.

2. Introduction (Continued)

- In this document we shall consider three kinds of entities:
  - ⋆ digital recordings of artistic and intellectual nature:
    - ◇ music, movies, readings ("audio books"), and the like,
  - ⋆ patients in a hospital
    - ◇ as represented also by their patient medical records, and
  - ⋆ documents, say of public government.

2. Introduction (Continued)

- The *permissions* and *obligations* issues are,
  - (1) for the owner (agent) of some intellectual property to be paid (an *obligation*) by users when they perform *permitted* operations on their works;
  - (2) for the patient to be professionally treated — by medical staff who are basically *obliged* to try to cure the patient; and
  - (3) for public administrators and citizens to enjoy good governance: transparency in law making (national parliaments and local prefecture and city councils), in law enforcement (i.e., the daily administration of laws), and law interpretation (the judiciary) — by agents who are basically *obliged* to produce certain documents while being *permitted* to consult (i.e., read) other documents.

2.1 What Kind of Science Is This?

- It is experimental computing science:
  - ⋆ The study and knowledge of how to design and construct
    - ◇ software that is right, i.e., correct,
    - ◇ and the right software, i.e., what the user wants.
  - ⋆ To study methods for getting the right software is interesting.
  - ⋆ To study methods for getting the software right is interesting.
  - ⋆ Domain development helps us getting the right software.
  - ⋆ Deriving requirements from domain descriptions likewise.
  - ⋆ Designing software from such requirements helps us get the software right.
- Understanding a domain and then designing license languages from such an understanding is new.
- We claim that computer-supported management of properly designed license languages is a hallmark of the e-Society.

2.2 What Kind of Contributions?

- The experimental nature of the project being reported on is as follows:
  - ⋆ Postulate three domains.
  - ⋆ Decribe these informally and formally.
  - ⋆ Postulate the possibility of license languages (LLs) that somehow relate to actitivities of respective domains.
  - ⋆ Design these – experimentally.
  - ⋆ Try discover similarities and differences between the three LLs (LL$_{DRM}$, LL$_{HHLL}$, LL$_{PALL}$).
  - ⋆ Formalise the common aspects: C_{LL}.
  - ⋆ Formalise the three LLs — while trying to “parameterise” the C_{LL} to achieve $LL_{DRM}, LL_{HHLL}, LL_{PALL}$.
- This investigation is bound to tell us something.
- It requires loads of work.

3. Pragmatics of The Three License Languages

- *By* **pragmatics** *we understand the*
  - ⋆ *study and*
  - ⋆ *practice*

  of
  - ⋆ *the factors that govern our choice of language in social interaction and*
  - ⋆ *the effects of our choice on others.*
- *We use the term pragmatics in connection with the use of language, as complemented by the semantics and syntax of language.*

3. Pragmatics of The Three License Languages (Continued)

- In this section we shall rough-sketch describe pragmatic aspects of the three domains of
  - ⋆ (1) production, distribution and consumption of artistic works,
  - ⋆ (2) the hospotalisation of patient, i.e., hospital healthcare and
  - ⋆ (3) the handling of law-based document in public government.
- The emphasis is on the pragmatics of the terms,
- i.e., the language used in these three domains.

3.1 The Performing Arts: Producers and Consumers

- The intrinsic entities of the performing arts are the artistic works:
  - ⋆ drama or opera performances,
  - ⋆ music performances,
  - ⋆ readings of poems, short stories, novels, or jokes,
  - ⋆ movies,
  - ⋆ documentaries, newsreels, etc.
- We shall limit our span to the scope of electronic renditions of these artistic works:
  - ⋆ videos, CDs or other.
- In this talk we shall not touch upon the technical issues of "downloading" (whether "streaming" or copying, or other). That and other issues will be analysed in a report by Dr. Xiang, JAIST.

3.1.1 Operations on Digital Works

- For a consumer to be able to enjoy these works that consumer must (normally first) usually "buy a ticket" to their performances.
- The consumer, i.e., the theatre, opera, concernt, etc., "goer" (usually)
  - ⋆ cannot copy the performance (e.g., "tape it"),
  - ⋆ let alone edit such copies of performances.
- In the context of electronic, i.e., digital renditions of these performances the above "cannots" take on a new meaning.
  - ⋆ The consumer may copy digital recordings,
  - ⋆ may edit these,
  - ⋆ and may further pass on such copies or editions to others.

3.1.1 Operations on Digital Works (Continued)

- To do so, while protecting the rights of the producers (owners, performers),
  - ⋆ the consumer requests permission to have the digital works transferred ("downloaded") from the owner/producer to the consumer,
  - ⋆ so that the consumer
    - ◇ can **render** ("play") these works on own rendering devices (CD, DVD, etc., players),
    - ◇ possibly can **copy** all or parts of them,
    - ◇ then possibly can **edit** all or parts of the copies, and,
    - ◇ finally, possibly can further **license** these "edited" versions to other consumers subject to payments to "original" licensor.

3.1.2 License Agreement and Obligation

- To be able to obtain these permissions
  - ⋆ the user agrees with some license
  - ⋆ and pays for the rights to operate on the digital works.

3.1.3 The Artistic Electronic Works: Two Assumptions

- Two, related assumptions underly the pragmatics of the electronics of the artistic works.
  - ⋆ The first assumption is that
    - ◇ the format, the electronic representation of the artistic works is proprietary,
    - ◇ that is, that the producer still owns that format.
  - ⋆ Either the format
    - ◇ is publicly known
    - ◇ or it is not, that is, it is somehow "secret".

3.1.3 The Artistic Electronic Works: Two Assumptions (Continued)

- In either case we "derive" the second assumption (from the fullfilment of the first).
  - ⋆ The second assumption is that the consumer is not allowed to, or cannot operate[2] on the works by own means (software, machines).
- The second assumption implies that
  - ⋆ acceptance of a license
  - ⋆ results in the consumer receiving software
  - ⋆ that supports the consumer in performing all operations on
  - ⋆ licensed works, their copies and edited versions:
    - ◇ rendering,
    - ◇ copying,
    - ◇ editing and
    - ◇ sublicensing.

[2] render, copy and edit

3.1.4 Protection of the Artistic Electronic Works

- The issue now is:
  - ⋆ how to protect the intellectual property
  - ⋆ (i.e., artistic) and financial (explotation) rights
  - ⋆ of the owners of the possibly rendered, copied and edited works,
  - ⋆ both when, and when not further distributed.

3.1.5 An Artistic Digital Rights License Language

- In this talk we shall
  - ⋆ design a suitably flexible digital artistics works license language
  - ⋆ and, through its precise informal and formal description
  - ⋆ provide one set answers to the above issue.

3.2 Hospital Healthcare: Patients and Medical Staff

- Citizens
  - ⋆ go to hospitals
  - ⋆ in order to be treated for some calamity (disease or other),
  - ⋆ and by doing so these citizens become patients.
- At hospitals patients, in a sense, issue a request to be cured.
- This request is directed at medical staff, that is,
  - ⋆ the patient authorises medical staff to perform a set of actions upon the patient.
  - ⋆ One could claim, as we shall, that the patient issues a license.

3.2.1 Hospital Healthcare: Patients and Patient Medical Records

- So patients and their attendant patient medical records (PMRs) are the main entities, the "works" of this domain.
- We shall treat them synonymously: PMRs as surrogates for patients.
- Typical actions on patients — and hence on PMRs — involve
  - ⋆ admitting patients,
  - ⋆ interviewing patients,
  - ⋆ analysing patients,
  - ⋆ diagnosing patients,
  - ⋆ planning treatment for patients,
  - ⋆ actually treating patients, and,
  - ⋆ under normal circumstance, to finally release patients.

3.2.2 Hospital Healthcare: Medical Staff

- Medical staff may request ('refer' to)
  - ⋆ other medical staff to perform some of these actions.
  - ⋆ One can conceive of describing action sequences (and 'refers') in the form of hospitalisation (not treatment) plans.
  - ⋆ We shall call such scripts for licenses.

3.2.3 Professional Healthcare

- The issue is now,
  - ⋆ given that we record these licenses,
  - ⋆ their being issued and being honoured,
  - ⋆ whether the handling of patients at hospitals
    - ◇ follow,
    - ◇ or does not follow

  properly issued licenses.

3.2.4 A Hospital Healthcare License Language

- In this talk we shall
  - ⋆ design a suitably flexible hospital healthcare license language
  - ⋆ and, through its precise informal and formal description
  - ⋆ provide one set answers to the above issue.

3.3 Public Government and the Citizens

3.3.1 The Three Branches of Government

- By public government we shall,
  - ⋆ following Charles de Secondat, baron de Montesquieu (1689–1755)[3],
  - ⋆ understand a composition of three powers:
    - ◇ the law making (legislative),
    - ◇ the law enforcing and
    - ◇ the law interpreting.

  parts of public government.

[3] *De l'esprit des lois* (*The Spirit of the Laws*), published 1748

3.3.1 The Three Branches of Government (Continued)

- Typically we refer to
  - ⋆ law making government as the national parliament and local (province and city) councils,
  - ⋆ law enforcing government as the executive (the administration),
  - ⋆ and law interpreting government as the judiciary [system] (including lawyers etc.).

3.3.2 Documents

- A crucial means of expressing public administration is through *documents.*[4]
- We shall therefore provide a brief domain analysis of a concept of documents.
- (This document domain description also applies
  - ⋆ to patient medical records and,
  - ⋆ by some "light" interpretation, also to artistic works —

  insofar as they also are documents.)

[4] Documents are, for the case of public government to be the "equivalent" of artistic works.

3.3.2 Documents (Continued)

- Documents are
  - ⋆ created,
  - ⋆ edited and
  - ⋆ read;
- and documents can be
  - ⋆ copied,
  - ⋆ distributed, the subject of
  - ⋆ calculations (interpretations) and be
  - ⋆ shared and
  - ⋆ shredded.

3.3.3 Document Attributes

- With documents one can associate, as attributes of documents, the actors who
  - ⋆ created,
  - ⋆ edited,
  - ⋆ read,
  - ⋆ distributed
  - ⋆ (to whom distributed),
  - ⋆ shared
  - ⋆ performed calculations on,
  - ⋆ shredded

  documents.

3.3.3 Document Attributes (Continued)

- With these operations on documents,
- and hence as attributes of documents one can, again conceptually,
- associate the
  - ⋆ location and
  - ⋆ time

  of these operations.

3.3.4 Actor Attributes and Licenses

- With actors (whether agents of public government or citizens)
  - ⋆ one can associate the authority (i.e., the rights)
  - ⋆ these actors have with respect to performing actions on documents.
- We now intend to express these authorisations as licenses.

3.3.5 Document Tracing

- An issue of public government is
  - ⋆ whether citizens and agents of public government act in accordance with the laws —
  - ⋆ with actions and laws reflected in documents
  - ⋆ such that the action documents enables a trace from the actions to the laws "governing" these actions.

- We shall therefore assume that every document can be traced
  - ⋆ back to its law origin
  - ⋆ as well as to all the documents any one document creation or editing was based on.

3.3.6 A Public Administration Document License Language

- In this talk we shall
  - ⋆ design a suitably flexible public government document license language
  - ⋆ and, through its precise informal and formal description
  - ⋆ provide one set answers to the above issue.

4. The Semantic Intents of Licensor and Licensee Actions

4.1 Overview

- There are two parties to a license:
  - ⋆ the licensor and
  - ⋆ the licensee.
- And there is a
  - ⋆ common agreement
  - ⋆ concerning a shared "item"

  between them, namely:
  - ⋆ the license and
  - ⋆ the "work", the patient, the document

- The licensor
  - ⋆ gives the licensee permission,
  - ⋆ or mandates the licensee to be obligated
- to perform certain actions
- on designated "items".

- The licensee
  - ⋆ performs, or
  - ⋆ does not perform
  - ⋆ permitted and/or
  - ⋆ obligated

  actions

- And the licensee
  - ⋆ may perform actions
  - ⋆ not permitted or
  - ⋆ not obligated.

- The license shall serve to
  - ⋆ ensure that only permitted actions are carried out, and to
  - ⋆ ensure that all obligated actions are carried out.

- Breach of the above, that is, breach of the contracted license
  - ⋆ may or shall result in
  - ⋆ revocation of the license.

4.2 Licenses and Actions

4.2.1 Licenses

- Conceptually a
  - ⋆ licensor o (for owner)
  - ⋆ may issue a license named ℓ
  - ⋆ to licensee u (for user)
  - ⋆ to perform some actions.
- The license may syntactically appear as follows:

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w

4.2.2 Actions

- And, conceptually, the licensee (u)
- may perform actions

  ℓ :a(w); ℓ :a′(w); ...; ℓ :a″(w); ...; ℓ :a‴(w)

- These actions ($a, a', ..., a'', ..., a'''$)
  - ⋆ may be in the set {a1,a2,...,an}
  - ⋆ or they may not be in the set.
- In the latter case we have a breach of license ℓ.

4.2.3 Two Languages

- Thus there is
  - ⋆ the language of licenses
  - ⋆ and the language of actions.

4.3 Sublicensing

- A licensee u may wish to delegate some of the work associated with performing some licensed actions to a colleague.
- If, for example the license originally stated:

  ℓ : **licensor** o **licenses licensee** u
  to perform actions {a1,a2,...,an} **on work item** w

  ⋆ the licensee (u) may wish a colleague u' to perform a subset

  of the actions, for example

  {ai,aj,...,ak} $\subseteq$ {a1,a2,...,an}

4.3 Sublicensing (Continued)

- Therefore u would like if the above license

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w

- instead was formulated as:

ℓ : **licensor** o **licenses licensee** u
to perform actions {a1,a2,...,an} **on work item** w
allowing sublicensing of actions {ai,aj,...,ak}

- where

{ai,aj,...,ak} $\subseteq$ {a1,a2,...,an}

4.3 Sublicensing (Continued)

- Now licensee u can perform the action

  ℓ : **license actions** {a′,a″,a‴} **to** u′

- The above is an action designator.
- It practical meaning is that a license is issued by u:

  $\eta(\ell$,u,t): **licensor** u **licenses licensee** u′
  to perform actions {a′,a″,a‴} **on work item** w

- The above license can be easily "assembled" from the action including the action named license:
  - ⋆ the context determines who (namely u) is issuing the license,
  - ⋆ and who or which is the work item.
- η is a function which applies to license name, agent identifications and time and yields unique new license names.

4.4 Multiple Licenses

- Conside the following scenario:
  - ⋆ A licensee L is performing actions $a_p, a_q, \ldots, a_r$,
  - ⋆ and has licensed L' to perform actions $a_i, a_j, \ldots, a_k$.
  - ⋆ The action licensing L' occurs at some time.
  - ⋆ At that time L has performed some of the actions $a_p, a_q, \ldots, a_r$, but maybe not all.
- What is going to happen?
  - ⋆ Can L and L' go on, in parallel, performing actions on the same work item?
  - ⋆ Our decision is yes, and they can do so in an interleaved manner,
  - ⋆ not concurrently but alternating, i.e., not accessing the same work item simultaneously.

5. Syntax and Informal Semantics

- We distinguish between
  - ⋆ the pragmatics,
  - ⋆ the semantics
  - ⋆ and the syntax

  of languages.

- *By* **syntax** *we mean (i) the ways in which words are arranged to show meaning (cf. semantics) within and between sentences, and (ii) rules for forming syntactically correct sentences.*

- *By* **semantics** *we mean the study and knowledge [incl. specification] of meaning in language .*

- By **informal** *semantics we mean a semantics which is expressed in concise natural language, for example, as here, English.*

5.1 General Artistic License Language

- We refer to the abstract syntax formalised below (that is, formulas 0.–10.).
- The work on the specific form of the syntac has been facilitated by the work reported by Xiang JenWen

5.1.1 The Syntax

- The syntax has two parts.
  - ⋆ One for licenses being issued by licensors.
  - ⋆ And one for the actions that licensees may wish to perform.

5.1.1.1 Licenses

type

0. Ln, Nm, W, S, PF
1. L = Ln × Lic
2. Lic == mkLic(licensor:Nm,licensee:Nm,wrk:W,cmds:Cmd-**set**,fees:PF)
3. Cmd = ({|**render,copy,edit**|} × S*) | SubLic
4. SubLic = {|**license**|} × Cmd-**set** × {|**on work**|} × V

The above syntax is correct RSL. But it is decorated! The subtypes {|**boldface keyword**|} are inserted for readability.

5.1.1.1.1 Annotation: Licenses, 0–4

- (0.) Licenses
  - ⋆ are given names,
    - ◇ ln:Ln,
    - ◇ so are actors (owners and users), nn:Nm.
  - ⋆ By w:W we mean a (net) reference to the downloaded artistic work being licensed,
  - ⋆ By s:S we mean a selector to either a downloaded work or to a copied and or edited work.
  - ⋆ By pf:PF we mean a further undefined payment feel function.

5.1.1.1.1 Annotation: Licenses, 0–4 (Continued)

- (1.) Every license (lic:Lic) has a unique name (ln:Ln).
- (2.) A license (lic:Lic) containes five parts:
  - ⋆ the name of the licensor,
  - ⋆ the name of the licensee,
  - ⋆ a reference to (the name of) the work,
  - ⋆ a set of commands (that may be permitted to be performed on the work),
  - ⋆ and a payment fee function (or table).

5.1.1.1.1 Annotation: Licenses, 0–4 (Continued)

- (3.) A command is
  - ⋆ either a render, a copy or an edit command,
  - ⋆ or a sublicensing

  command.
- The render, copy and edit commands are each "decorated" with a an ordered list of selectors (i.e., selector names).
  - ⋆ The license command

    copy $\langle$s1,s2,s7$\rangle$

  - ⋆ means that the licensed work, w, may have its sections s1, s2 and s7 copied, and in that sequence.
  - ⋆ Other copy commands may specify other sequences.
- Similarly for render and edit commands.

5.1.1.1.1 Annotation: Licenses, 0–4 (Continued)

- The license command

  license cmd1,cmd2,...,cmdn **on work** v

  means

  - that the licensee is allowed to act as a licensor,
  - to name sublicensees (that is, licensees) freely,
  - to select only a (possibly full) subset of the sublicensed commands (that are listed) for the sublicensee to enjoy.

- The license need thus not mention the name(s) of the possible sublicensees.
  - But one could design a license language, i.e., modify the present one to reflect such constraits.

- The license also do not mention the payment fee component.
  - As we shall see under licensor actions such a function will eventually be inserted.

5.1.1.1.2 Annotation: Licenses, 0–4, General Remarks

- A license licenses the licensee to render, copy, edit and license (the results of) any selection of the downloaded work.
- In any order — but see below — and any number of times.
- For every time any of these operations take place payment according to the payment function, PF, occurs.
- The user can render the downloaded work (W) and can render copies of the work as well as edited versions of these (V).
- Editing is always of copied versions.
- Copying is either of downloaded or of copied or edited versions.
- This does not transpire from the license syntax but is expressed by the licensee, see below,
- and can be checked and duly paid for according to the payment function PF.

5.1.1.1.2 Annotation: Licenses, 0–4, General Remarks (Continued)

- The payment function is considered a partial function of the selections of the work being licensed.
- Please recall that licensed works are proprietary.
  - ⋆ Either the work format is known, or it is not supposed to be known,
  - ⋆ In any case,
    - ◇ the rendering, editing, copying and the license assembling (more later) functions
    - ◇ "follow" with the license
    - ◇ and are also assumed to be proprietary.
  - ⋆ Thus the licensee is not allowed to and may not be able to use own software for rendering, editing, copying and license assemblage.

5.1.1.1.2 Annotation: Licenses, 0–4, General Remarks (Continued)

- Licenses specify sets of permitted actions.
- Not sequences.
- Of course, the licensee, assumed to be an un-cloned human, can only perform one action at a time.
- So licensed actions are carried out sequentially.
- The order is not prescribed, but is decided upon by the licensee.
- Of course, some actions must precede other actions.
  - ⋆ Licensees must copy before they can edit,
  - ⋆ and they usually must edit some copied work before they can sublicense it.
  - ⋆ But the latter is strictly speaking not necessary.

5.1.1.2 Actions

type

5. V
6. Act = Ln × (Rndr|Copy|Edit|License)
7. Rndr == mkR(sel:S*,wrk:(W|V))
8. Copy == mkC(sel:S*,wrk:(W|V),into:V)
9. Edit == mkE(wrks:V*,into:V)
10. License == mkL(ln:Ln,licensee:Nm,wrk:V,cmds:Cmd-**set**,fees:PF)

5.1.1.2.1 Annotation: Licensor Actions, 5–10

- (5.) By V we mean the name of a variable in the users own storage into which downloaded works can be copied (now becoming a local work.
  - ⋆ The variables are not declared.
  - ⋆ They become defined when the licensee names them in a copy command.
  - ⋆ They can be overwritten.
  - ⋆ No type system is suggested.

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

- (6.) Every action of a licensee is tagged by the name of a relevant license.
  - ⋆ If the action is not authorised by the named license then it is rejected.
  - ⋆ Render and copy actions mention a specific sequence of selectors.
  - ⋆ If this sequence is not an allowed (a licensed) one, then the action is rejected.
  - ⋆ (Notice that the license may authorise a specific action, a with different sets of sequences of selectors — thus allowing for a variety of possibilities as well as constraints.)

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

- (7.) The licensee, having now received a license,
  - ⋆ can **render** selections
    - ◇ of the licensed work,
    - ◇ or of copied
    - ◇ and/or edited versions

    of the licensed work.
  - ⋆ No reference is made to the payment function.
    - ◇ When rendering the semantics is that this function is invoked and duly applied.
    - ◇ That is, render payments are automatically made: subtracted from the licensees account and forwarded to the licensor.

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

- (8.) The licensee
  - ⋆ can **copy** selections
    - ◇ of the licensed work,
    - ◇ or of previously copied
    - ◇ and/or edited versions

    of the licensed work.
  - ⋆ The licensee identifies a name for the local storage file where the copy will be kept.
  - ⋆ No reference is made to the payment function.
    - ◇ When copying the semantics is that this function is invoked and duly applied.
    - ◇ That is, copy payments are automatically made: subtracted from the licensees account and forwarded to the licensor.

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

- (9.) The licensee
  - ⋆ can **edit** selections
    - ◇ of the licensed work,
    - ◇ or of copied
    - ◇ and/or previously edited versions

    of the licensed work.
  - ⋆ The licensee identifies a name for the local storage file where the new edited version will be kept.
  - ⋆ The result of editing is a new work.

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

- ⋆ No reference is made to the **payment** function.
  - ⋄ When copying the semantics is that this function is invoked and duly applied.
  - ⋄ That is, copy payments are automatically made: subtracted from the licensees account and forwarded to the licensor.
- ⋆ Although no reference is made to any edit functions these are made available to the licensee when invoking the edit command.
  - ⋄ You may think of these edit functions being downloaded at the time of downloading the license.
  - ⋄ Other than this we need not further specify the editing functions.
- ⋆ Same remarks apply to the above copying functions.

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

- (10.) The licensee can further **sublicense** copied and/or edited work.
  - ⋆ The licensee must give the license being assembled a unique name.
  - ⋆ And the licensee must choose to whom to license this work.
  - ⋆ A sublicense, like does a license, authorises which actions can be performed, and then with which one of a set of alternative selection sequences.
  - ⋆ No payment function is explicitly mentioned.
    - ◇ It is to be semi-automatically derived
    - ◇ (from the originally licensed payment fee function
    - ◇ and the licensee's payment demands)
    - ◇ by means of functionalities provided as part of the licensed payment fee function.

5.1.1.2.1 Annotation: Licensor Actions, 5–10 (Continued)

⋆ The command information is thus **compiled** (**assembled**) into a license of the form given in (1.–3.).

◇ The licensee becomes the licensor and the recipient of the new, the sublicense, become the new licensee.

◇ The assemblage refers to the context of the action.

◇ That context knows who, the licensor, is issuing the sublicense.

◇ From the license label of the command it is known whether the sublicense actions are a subset of those for which sublicensing has been permitted.

5.2 Hospital Healthcare License Language

5.2.1 A Notion of License Execution State

- In the context of the Artistic License Language licensees could basically perform licensed actions in any sequence and as often as they so desired.
  - ⋆ There were, of course, some obvious constraints.
    - ◇ Operations on local works could not be done before these had been created — say by copying.
    - ◇ Editing could only be done on local works and hence required a prior action of, for example, copying a licensed work.
- In the context of hospital health care most of the actions can only be performed if the patient has reached a suitable state in the hospitalisation.
- We refer to Fig. 1 for an idealised hospitalisation plan.

5.2.1 A Notion of License Execution State (Continued)

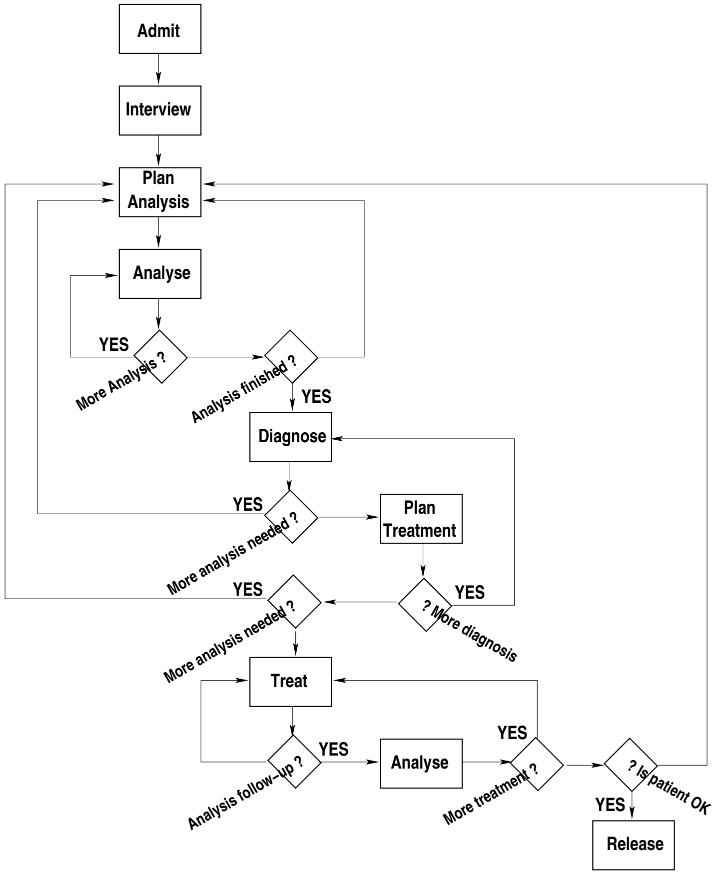

Figure 1: An example hospitalisation plan

5.2.1 A Notion of License Execution State (Continued)

- We therefore suggest
  - ⋆ to join to the licensed commands
  - ⋆ an indicator which prescribe the state of the hospitalisation plan in which the command action may be performed.
- Two or more medical staff may now be licensed
  - ⋆ to perform different actions
  - ⋆ in same or different states.
  - ⋆ If licensed to perform same action(s) in same state(s) —
  - ⋆ well that may be "bad license programing"!
- One cannot design a language and prevent it being misused!

5.2.2 The Syntax

- The syntax has two parts.
  - ⋆ One for licenses being issued by licensors.
  - ⋆ And one for the actions that licensees may wish to perform.

5.2.2.1 Licenses

type

0. Ln, Mn, Pn
1. License = Ln × Lic
2. Lic == mkLic(staff1:Mn,mandate:ML,pat:Pn)
3. ML == mkML(staff2:Mn,to_perform_acts:CoL-**set**)
4 CoL = Cmd | ML | Alt
5. Cmd == mkCmd(σ:Σ,stmt:Stmt)
6 Alt == mkAlt(cmds:Cmd-**set**)
7. Stmt = **admit** | **interview** | **plan-analysis** | **do-analysis** | **diagnose** | **plan-treatment** | **treat** | **transfer** | **release**

- The above syntax is correct RSL .
- But it is decorated!
- The subtypes {|**boldface keyword**|} are inserted for readability.

5.2.2.1.1 Annotations: Licenses

- (0.) Licenses, medical staff and patients have names.
- (1.) Licenses further consist of license bodies (Lic).
- (2.) A license body names the licensee (Mn), the patient (Pn), and,
- (3.) through the "mandated" licence paert (ML), it names the licensor (Mn) and which set of commands (C) or (o) implicit licenses (L, for CoL) the licensor is mandated to issue.
- (4.) An explicit command or licensing (CoL) is either a command (Cmd), or a sublicense (ML) or an alternative.
- (5.) A command (Cmd) is a state-labelled statement.

5.2.2.1.1 Annotations: Licenses (Continued)

- (ML, 3.) A sublicense just states the command set that the sublicense licenses.
  - ⋆ As for the Artistic License Language the licensee
  - ⋆ chooses an appropriate subset of commands.
  - ⋆ The context "inherits" the name of the patient.
  - ⋆ But the sublicensee is explicitly mandated in the license!
- (6.) An alternative is also just a set of commands.
  - ⋆ The meaning is that
    - ◇ either the licensee choose to perform the designated actions
    - ◇ or, as for ML, but now freely choosing the sublicensee,
    - ◇ the licensee (now new licensor) chooses to confer actions to other staff.

5.2.2.1.1 Annotations: Licenses (Continued)

- (7.) A statement is either
  - ⋆ an admit,
  - ⋆ an interview,
  - ⋆ a plan analysis,
  - ⋆ an analysis,
  - ⋆ a diagnose,
  - ⋆ a plan treatment,
  - ⋆ a treatment,
  - ⋆ a transfer, or
  - ⋆ a release

  directive
- Information given in the patient medical report
  - ⋆ for the designated state
  - ⋆ inform medical staff as to the details
  - ⋆ of analysis, what to base a diagnosis on, of treatment, etc.

5.2.2.2 Actions

8. Action = Ln × Act
9. Act = Stmt | SubLic
10. SubLic = mkSubLic(sublicensee:Ln,license:ML)

5.2.2.2.1 Annotations: Actions

- (8.) Each action actually attempted by a medical staff refers to the license, and hence the patient name.
- (9.) Actions are either of
  - ⋆ an admit,
  - ⋆ an interview,
  - ⋆ a plan analysis,
  - ⋆ an analysis,
  - ⋆ a diagnose,
  - ⋆ a plan treatment,
  - ⋆ a treatment,
  - ⋆ a transfer, or
  - ⋆ a release

  actions.

5.2.2.2.1 Annotations: Actions (Continued)

- Each individual action is only allowed in a state σ
  - ⋆ if the action directive appears in the named license
  - ⋆ and the patient (medical record) designates state σ.
- (10.) Or an action can be a sublicensing action.
  - ⋆ Either the sublicensing action that the licensee is attemting is explicitly mandated by the license (4. ML),
  - ⋆ or is an alternative one thus implicitly mandated (6.).
  - ⋆ The full sublicense, as defined in (1.–3.) is compiled from contextual information.

5.3 Public Administration License Language

5.3.1 The Syntax: Licenses

- The syntax has two parts.
  - ⋆ One for licenses being issued by licensors.
  - ⋆ And one for the actions that licensees may wish to perform.

5.3.1.1 The Form of Licenses

type

0. Ln, An, Cfn
1. L == Grant | Extend | Restrict | Withdraw
2. Grant == mkG(license:Ln,licensor:An,granted_ops:Op-**set**,licensee:An
3. Extend == mkE(licensor:An,licensee:An,license:Ln,with_ops:Op-**set**)
4. Restrict == mkR(licensor:An,licensee:An,license:Ln,to_ops:Op-**set**)
5. Withdraw == mkW(licensor:An,licensee:An,license:Ln)
6. Op == Crea|Edit|Read|Copy|Licn|Shar|Rvok|Rlea|Rtur|Calc|Shrd

5.3.1.2 Licensed Operations

type

7. Dn, DCn, UDI
8. Crea == mkCr(dn:Dn,doc_class:DCn,based_on:UDI-**set**)
9. Edit == mkEd(doc:UDI,based_on:UDI-**set**)
10. Read == mkRd(doc:UDI)
11. Copy == mkCp(doc:UDI)
12a. Licn == mkLi(kind:LiTy)
12b. LiTy == grant | extend | restrict | withdraw
13. Shar == mkSh(doc:UDI,with:An-**set**)
14. Rvok == mkRv(doc:UDI,from:An-**set**)
15. Rlea == mkRl(dn:Dn)
16. Rtur == mkRt(dn:Dn)
17. Calc == mkCa(fcts:CFn-**set**,docs:UDI-**set**)
18. Shrd == mkSh(doc:UDI)

5.3.1.3 Annotations: Licenses

- (0.) The are names of licenses (Ln), actors (An), documents (UDI), document classes (DCn) and calculation functions (Cfn).
- (1.) There are four kinds of licenses: granting, extending, restrictiing and withdrawing.
- (2.) Actors (licensors) grant licenses to other actors (licensees).
  - ⋆ An actor is constrained to always grant distinctly named licenses.
  - ⋆ No two actors grant identically named licenses.[5]
  - ⋆ A set of operations on (named) documents are granted.

[5] This constraint is securable by letting the actor name be part of the license name.

5.3.1.3 Annotations: Licenses (Continued)

- (3.–5.) Actors who have issued named licenses may extend, restrict or withdraw the license rights (wrt. operations, or fully).
- (6.) There are nine kinds of operation authorisations. Some of the next explications also explain parts of some of the corresponding actions (see (16.–24.).
- (7.) There are names of documents (Dn), names of classes of documents (DCn), and there are unique document identifiers (UDI).

5.3.1.3 Annotations: Licenses (Continued)

- (8.) **Creation** results in an initially void document which is
  - ⋆ not necessarily uniquely named (dn:Dn) (but that name is uniquely associated with the unique document identifier created when the document is created[6])
  - ⋆ typed by a document class name (dcn:DCn) and possibly
  - ⋆ based on one or more identified documents (over which the licensee (at least) has reading rights).
  - ⋆ We can presently omit consideration of the document class concept.
  - ⋆ "based on" means that the initially void document contains references to those (zero, one or more) documents.[7]
  - ⋆ The "based on" documents are moved from licensor to licensee.

[6]— hence there is an assumption here that the create operation is invoked by the licensee exactly (or at most) once.
[7]They can therefore be traced.

5.3.1.3 Annotations: Licenses (Continued)

- (9.) **Editing** a document
  - ⋆ may be based on "inspiration" from, that is, with reference to a number of other documents (over which the licensee (at least) has reading rights).
  - ⋆ What this "be based on" means is simply that the edited document contains those references. (They can therefore be traced.)
  - ⋆ The "based on" documents are moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions.
- (10.) **Reading** a document
  - ⋆ only changes its "having been read" status.
  - ⋆ The read document, if not the result of a copy, is moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions.

5.3.1.3 Annotations: Licenses (Continued)

- (11.) **Copying** a document
  - ⋆ increases the document population by exactly one document.
  - ⋆ All previously existing documents remain unchanged except that the document which served as a master for the copy has been so marked.
  - ⋆ The copied document is like the master document except that the copied document is marked to be a copy.
  - ⋆ The master document, if not the result of a create or copy, is moved from licensor to licensee
  - ⋆ — if not already so moved as the result of the specification of other authorised actions.

5.3.1.3 Annotations: Licenses (Continued)

- (12a.) A licensee can **sublicense** (sL) certain operations to be performed by other actors.
- (12b.) The granting, extending, restricting or withdrawing permissions,
  - ⋆ cannot name a license (the user has to do that),
  - ⋆ do not need to refer to the licensor (the licensee issuing the sublicense),
  - ⋆ and leaves it open to the licensor to freely choose a licensee.
  - ⋆ The licensor (the licensee issuing the sublicense) must choose a unique license name.

5.3.1.3 Annotations: Licenses (Continued)

- (13.) A document can be **shared**
  - ⋆ beween two or more actors.
  - ⋆ One of these is the licensee, the others are implicitly given read authoristations.
  - ⋆ (One could think of extending, instead the licensing actions with a **shared** attribute.)
  - ⋆ The shared document, if not the result of a create and edit or copy, is moved from licensor to licensee — if not already so moved as the result of the specification of other authorised actions.
  - ⋆ Sharing a document does not move nor copy it.

5.3.1.3 Annotations: Licenses (Continued)

- (14.) Sharing documents can be **revoked**. That is, the reading rights are removed.
- (15.) The **release** operation:
  - ⋆ if a licensor has authorised a licensee to create a document
  - ⋆ (and that document, when created got the unique document identifier udi:UDI)
  - ⋆ then that licensee can **release** the created, and possibly edited document (by that identification)
  - ⋆ to the licensor, say, for comments.
  - ⋆ The licensor thus obtains the master copy.

5.3.1.3 Annotations: Licenses (Continued)

- (16.) The **return** operation:
  - ⋆ if a licensor has authorised a licensee to create a document
  - ⋆ (and that document, when created got the unique document identifier udi:UDI)
  - ⋆ then that licensee can **return** the created, and possibly edited document (by that identification)
  - ⋆ to the licensor — "for good"!
  - ⋆ The licensee relinguiches all control over that document.

5.3.1.3 Annotations: Licenses (Continued)

- (17.) Two or more documents can be subjected to any one of a set of permitted **calculation** functions.
  - ⋆ These documents, if not the result of a creates and edits or copies, are moved from licensor to licensee —
  - ⋆ if not already so moved as the result of the specification of other authorised actions.
  - ⋆ Observe that there can be many calculation permissions, over overlapping documents and functions.
- (18.) A document can be **shredded**.
  - ⋆ It seems pointless to shred a document if that was the only right granted wrt. document.

5.3.2 The Syntax: Actions

17. Action = Ln × Clause
18. Clause = Cre | Edt | Rea | Cop | Lic | Sha | Rvk | Rel | Ret | Cal | Shr
19. Cre == mkCre(dcn:DCn,based_on_docs:UID-**set**)
20. Edt == mkEdt(uid:UID,based_on_docs:UID-**set**)
21. Rea == mkRea(uid:UID)
22. Cop == mkCop(uid:UID)
23. Lic == mkLic(license:L)
24. Sha == mkSha(uid:UID,with:An-**set**)
25. Rvk == mkRvk(uid:UID,from:An-**set**)
25. Rev == mkRev(uid:UID,from:An-**set**)
26. Rel == mkRel(dn:Dn,uid:UID)
27. Ret == mkRet(dn:Dn,uid:UID)
28. Cal == mkCal(fct:Cfn,over_docs:UID-**set**)
29. Shr == mkShr(uid:UID)

5.3.2.1 Preliminary Remarks

- A clause elaborates to a state change and usually some value.
- The value yielded by elaboration of the above
  - ⋆ create, copy, and calculation

  clauses
- are **unique document identifiers**.
- These are chosen by the "system".

5.3.2.2 Annotations: Actions

- (17.) Actions are **tagged** by the name of the license
  - ⋆ with respect to which their authorisation and document names has to be checked.
  - ⋆ No action can be performed by a licensee
  - ⋆ unless it is so authorised by the named license,
  - ⋆ both as concerns the operation (create, edit, read, copy, license, share, revoke, calculate and shred)
  - ⋆ and the documents actually named in the action.
  - ⋆ They must have been mentioned in the license,
  - ⋆ or, created or copies of downloaded (and possibly edited) documents or copies of these — in which cases operations are inherited.

5.3.2.2 Annotations: Actions (Continued)

- (19.) A licensee may **create** documents if so licensed —
  - ⋆ and obtains all operation authorisations to this document.
- (20.) A licensee may **edit** "downloaded" (edited and/or copied) or created documents.
- (21.) A licensee may **read** "downloaded" (edited and/or copied) or created and edited documents.
- (22.) A licensee may (conditionally) **copy** "downloaded" (edited and/or copied) or created and edited documents.
  - ⋆ The licensee decides which name to give the new document, i.e., the copy.
  - ⋆ All rights of the master are inherited to the copy.

5.3.2.2 Annotations: Actions (Continued)

- (23.) A licensee may **issue licenses**
  - ⋆ of the kind permitted.
  - ⋆ The licensee decides whether to do so or not.
  - ⋆ The licensee decides
    - ◇ to whom,
    - ◇ over which, if any, documents,
    - ◇ and for which operations.
  - ⋆ The licensee looks after a proper ordering of licensing commands:
    - ◇ first grant,
    - ◇ then sequences of zero, one or more either extensions or restrictions,
    - ◇ and finally, perhaops, a withdrawal.

5.3.2.2 Annotations: Actions (Continued)

- (24.) A "downloaded" (possibly edited or copied) document may (conditionally) be **shared** with one or more other actors.
  - ⋆ Sharing, in a digital world, for example,
  - ⋆ means that any edits done after the opening of the sharing session,
  - ⋆ can be read by all so-granted other actors.
- (25.) Sharings may (conditionally) be **revoked**, partially or fully, that is, wrt. original "sharers".

5.3.2.2 Annotations: Actions (Continued)

- (26.) A document may be **released**.
  - ⋆ It means that the licensor who originally requested
  - ⋆ a document (named dn:Dn) to be created
  - ⋆ now is being able to see the results —
  - ⋆ and is expected to comment on this document
  - ⋆ and eventually to re-license the licensee to further work.
- (27.) A document may be **returned**.
  - ⋆ It means that the licensor who originally requested
  - ⋆ a document (named dn:Dn) to be created
  - ⋆ is now given back the full control over this document.
  - ⋆ The licensee will no longer operate on it.

5.3.2.2 Annotations: Actions (Continued)

- (28.) A license may (conditionally) apply any of a licensed set of **calculation functions**
  - ⋆ to "downloaded" (edited, copied, etc.) documents,
  - ⋆ or can (unconditioally) apply any of a licensed set of calculation functions
  - ⋆ to created (etc.) documents.
  - ⋆ The result of a calculation is a document.
  - ⋆ The licensee obtains all operation authorisations to this document (— as for created documents).
- (29.) A license may (conditionally) **shred** a "downloaded" (etc.) document.

5.4 Discussion

5.4.1 Comparisons

- We have “designed”, or rather proposed three different kinds of license languages.
  - ⋆ In which ways are they “similar”,
  - ⋆ and in which ways are they “different”?
- Proper answers to these questions can only bew given after we have formalised these languages.
  - ⋆ The comparisons can be properly founded on
  - ⋆ comparing the semantic values of these languages.

5.4.1 Comparisons (Continued)

- But before we embark on such formalisations
  - ⋆ we need some informal comparisons
  - ⋆ so as to guide our formalisations.
- So we shall attempt such analysis now with the understanding that it is only a temprary one.

5.4.1.1 Differences

5.4.1.1.1 Work Items

- The **work items** of the **artistic** license language(s) are essentially "kept" by the licensor.
- The **work items** of the **hospital healthcare** license language(s) are fixed and, for a largse set of licenses there is one work item, the patient which is shared between many licensors and licenses.
- The **work items** of the **public administration** license language(s) — namely document — are distributed to or created and copied by licenses and may possibly be shared.

5.4.1.1 Differences (Continued)
5.4.1.1.2 Operations

- The **operations** of the **artistic** license language(s) are are essentially “kept” by the licensor.
- The **operations** of the **hospital healthcare** license language(s) are are essentially “kept” by the licensees (as reflected in their professional training and conduct).
- The **operations** of the **public administration** license language(s) are essentially “kept” by the licensees (as reflected in their professional training and conduct).

5.4.1.1 Differences (Continued)
5.4.1.1.3 Permissions and Obligations

- Generally we can say that
  - ⋆ the **modalities** of the **artistic** license language(s) are essentially **permissions** with **payment** (as well as use of licensor functions) being an **obligation**;
  - ⋆ that the **modalities** of the **hospital healthcare** license language(s) are are essentially **obligations**;
  - ⋆ and, as well, that the **modalities** of the **public administration** license language(s) are essentially **obligations**
- We shall have more to say about permissions and obligations later (much later).

5.4.1.1 Differences (Continued)

5.4.1.1.4 Infringement

-
-
-
-

5.4.2 "Other"

-
-
-
-

5. Formal Semantics

- By formal semantics we understand a definition expressed in a formal language, that is, a language with
  - ⋆ a mathematical syntax,
  - ⋆ a matematical semantics,
  - ⋆ and a consistent and relative complete proof system.
- We shall deploy the CSP Specification Language embedded in a Landin–like notation of **let** clauses[8].
- We hope someone will complement our definition with a commensurate CafeOBJ definition.

[8]— known since the very early 1960's

5.1 A Model of Common Aspects

5.1.1 Actors: Behaviours and Processes

- We see the system as a set of actors.
- An actor is either a
  - ⋆ licensor, or a
  - ⋆ licensee, or,
    - ◇ usually, such as we have envisaged our license languages,
  - ⋆ both.
- To each actor we associate a behaviour —
  - ⋆ and we model actor behaviours as CSP processes.

5.1.1 Actors: Behaviours and Processes (Continued)

- So the system is then modelled
  - ⋆ as a set of concurrent behaviours,
    - ◇ that is, parallel ($\|$) CSP processes.
- Actors are uniquely identified (Aid).

5.1.1.1 System States

- With each actor behaviour we associate a state ($\omega : \Omega$).
  - ⋆ "Globally" initial such state components are modelled
    - ◇ as maps
    - ◇ from actor indentifiers
    - ◇ to states.
- We shall later analyse these states into major components.

type

  Aid, Ω

  $\Omega\mathrm{s} = \mathrm{Aid} \xrightarrow[m]{} \Omega$

5.1.1.2 System Processes

- Actor processes communicate with one another over channels.
- There is a set of actor identifier indexed channels.
- Potential licensees request licenses.
- Licensors issue licenses in response to requests.
- Work items are communicated over these channels.
- As are payments and reports on use of licensed operations on licensed work items.
- An actor is
  - ⋆ either pro-active, requesting licenses, sending payment or reports,
  - ⋆ or re-active: responding to license requests, sending work items.
- An actor non-deterministically ($\lceil\rceil$) alternatese between these activities.

5.1.1.2 System Processes (Continued)

type

$\mathrm{M} = \mathrm{Lic} \mid \mathrm{Pay} \mid \mathrm{Rpt} \mid \ldots$

channel

$\{\mathrm{a}[\,\mathrm{i}\,] | \mathrm{i}{:}\mathrm{Aid}\}\ (\mathrm{Aid} \times \mathrm{M})$

value

$\mathrm{actor}{:}\ \mathrm{j}{:}\mathrm{Aid} \times \Omega \to \textbf{in},\textbf{out}\ \{\mathrm{a}[\,\mathrm{i}\,] | \mathrm{i}{:}\mathrm{Aid}\cdot\mathrm{i} \neq \mathrm{j}\}\ \textbf{Unit}$

$\mathrm{actor}(\mathrm{j})(\omega) \equiv \textbf{let}\ \omega' = \mathrm{pro\_act}(\mathrm{j})(\omega) \sqcap \mathrm{re\_act}(\mathrm{j})(\omega)\ \textbf{in}\ \mathrm{actor}(\mathrm{j})(\omega')\ \textbf{end}$

$\mathrm{system}{:}\ \Omega\mathrm{s} \to \textbf{Unit}$

$\mathrm{system}(\omega\mathrm{s}) \equiv \|\ \{\mathrm{actor}(\mathrm{i})(\omega\mathrm{s}(\mathrm{i})) | \mathrm{i}{:}\mathrm{Aid}\}$

5.1.1.3 Actor Processes

- We have identified two kinds of actor processes:
  - ⋆ pro-active, during which the actor, by own initiative,
    - ◇ (as a prespective licensee) may request a license from a prospective licensor,
    - ◇ or, (as an actual licensee) as the result of performing licensed actions sends payments or reports (or other) to the licensor.
  - ⋆ and re-active, during which the actor, in response to requests (as a licensor)
    - ◇ sends a requesting actor a license (whereby the requestor becomes a licensee),
    - ◇ or "downloads" (access to) requested works or functions. functions.

5.1.1.3.1 The Pro-active Actor Behaviour

```
type
  M = Lic | Pay | Rpt | ...
channel
  {a[ i ]|i:Aid} (Aid×M)
value
  pro_act: j:Aid → Ω → in,out {a[ i ]|i:Aid•j≠i}  Ω
  pro_act(j)(ω) ≡
    let what_to_do = rl ⌈⌉ op in
    case what_to_do of
      rl → let (k,l_rq,ω′)=iq_l_Ω(ω) in a(aid)!(j,l_rq);ω′ end
      op → let op_i=iq_a_Ω(ω) in let (k,m,ω′)=act(op_i)(ω) in a(k)!(j,m) ; ω′ end end
    end end
```

5.1.1.3.2 The Re-active Actor Behaviour

type

$\mathrm{M} = \mathrm{Lic} \mid \mathrm{Pay} \mid \mathrm{Rpt} \mid \ldots$

channel

$\{\mathrm{a}[\,\mathrm{i}\,] | \mathrm{i}{:}\mathrm{Aid}\}\ (\mathrm{Aid} \times \mathrm{M})$

value

$\mathrm{re\_act}\colon\ \mathrm{j}{:}\mathrm{Aid} \rightarrow \Omega \rightarrow$ **in**,**out** $\{\mathrm{a}[\,\mathrm{i}\,] | \mathrm{i}{:}\mathrm{Aid}{\bullet}\mathrm{j}{\neq}\mathrm{i}\}\ \ \Omega$

$\mathrm{re\_act(j)}(\omega) \equiv$ **let** $(\mathrm{k,m}) = \lceil\!\rceil\{\mathrm{a(k)?} | \mathrm{k}{:}\mathrm{Aid}\}$ **in let** $(\mathrm{m}',\omega') = \mathrm{act(m)}(\omega)$ **in** $\mathrm{a(k)!(j,m')};\omega'$ **end en**

5.1.2 Functions

- We first list (and "read") the signatures of the
  - ⋆ two auxiliary (iq_l_Ω, iq_a_Ω) and
  - ⋆ one elaboration (act) function
  - ⋆ assumed in the definition of the pro- and re-active actor proesses.
- After that we discuss the former and suggest definitions of the latter.

5.1.2.1 Auxiliary Function Signatures

- The inquire license function (iq_l_Ω) inspects the actor's state to ("eureka") determine which most desirable licensor (Aid) offers which one kind of desired license requirements (License_Requirements).
- The inquire action function (iq_a_Ω) inspects the actor's state to (somewhat "eureka") determine which action is "next" in tine to be performed. That action is being designated (Action_Designator).

type
License_Requirements,Action_Designation
value
iq_l_Ω: $\Omega \rightarrow$ Aid $\times$ License_Requirements $\times\ \Omega$
iq_a_Ω: $\Omega \rightarrow$ Action_Designator

5.1.2.1 Auxiliary Function Signatures (Continued)

- By 'eureka'[9] is meant that the inquiry is internal non-deterministic
- that is, is not influenced by an outside,
- could have any one of very many outcomes,
- and can thus only be rather loosely defined.

[9] "Eureka" used to express triumph on a discovery, heuristics

5.1.2.2 Elaboration Function Signature

- The action performing function (act)
  - ⋆ "finds" the designated operation in the current state,
  - ⋆ applies it in the current state,
  - ⋆ and yields ("read" backwards)
    - ◇ a possibly new state ($\omega : \Omega$),
    - ◇ a message (M) to be sent to the
    - ◇ licensor (Aid)

    who

    - ◇ authorised the operation
    - ◇ and may need or wich to have a payment, a report, or some such thing "back"!

type
  Action_Designation
value
  act: Action_Designation $\rightarrow \Omega \rightarrow$ Aid $\times$ M $\times$ Ω

5.1.2.3 Discussion of Auxiliary Functions

- The auxiliary functions are usually not computable functions.
- The actors are not robots.
- And it is not necessary to further define these functions
  - ⋆ beyond stating their signatures
  - ⋆ as they are usually performed by human actors.
- The signature of the inquire license function expresses a possible change to the inquired state.
  - ⋆ One would think of the inquiring actor somehow noting down,
  - ⋆ remembering, aqs it were, which inquieries were attempted or had been made.
- The signature of the inquire actions function does not express such a state change.
  - ⋆ But it could be expressed as well.

5.1.2.4 Schema Definitions of Elaboration Functions

-
-
-

5.2 A Model of The General Artistic License Language

5.2.1 A Licensor/Licensee State

-
-
-

5.2.2 Auxiliary Functions

5.2.3 Elaboration Functions

-
-
-
-
-
-

5.3 A Model of The Hospital Healthcare License Language

5.3.1 A Patient [Medical Record] State

-
-
-
-
-
-

5.3.2 A Medical Staff State

-
-
-
-
-
-

5.3.3 Auxiliary Functions

-
-
-
-
-
-

5.3.4 Elaboration Functions

-
-
-
-
-
-

5.4 A Model of The Public Administration License Language

5.4.1 A Public Administrator State

-
-
-
-
-
-

5.4.2 Auxiliary Functions

-
-
-
-
-
-

5.4.3 Elaboration Functions

5.5 Discussion

6. Conclusion

- It is too early — in the development of this report — to conclude!

6.1 Summary: What Have We Achieved?

- Or rather, at this early, incomplete stage, what do we wish to achieve?
- In a first round we wish to achive the following:
  - ⋆ An understanding of different kinds of license languages.
  - ⋆ An understanding of obligations and permissions (yet to be "designed" more explicitly into the three languages.
  - ⋆ Formalisation of both common aspects of the license systems (as a "vastly" distributed set of very many actors acting on even more licenses "competing" for resources, etc.) as well as each individual language.
    - ◇ Both in RSL, being done by DB
    - ◇ and in CafeOBJ, to be done by XJW, CXY and AY respectively, as supported by OK.
  - ⋆ Lay a foundation for considerations of game-theoretic nature.

- In a second round we wish to achive the following:
  - ⋆ Clean up the understanding, the design and the formalisation of the languages.
  - ⋆ See broader aspects of license languages — beyond obligations and permissions.
  - ⋆ Work out, hopefully, with RV, more exiting aspects: Game-theoretic and otherwise!

6.2 Open Issues

$\boxed{\text{TO BE WRITTEN}}$

6.3 Acknowledgements

- The second author is happy to acknowledge the great pleasure it has been to work with his three co-author "students":
  - ⋆ Yasuhito Arimoto, MSc student at JAIST,
  - ⋆ Chen Xiaoyi, PhD student at JAIST, and
  - ⋆ Xinag Jianwen, "post-doc." at JAIST