A Family of License Languages

Incomplete R&D Development Notes[1], January 4, 2008, 16:51

Arimoto Yasuhito[2], Dines Bjørner[1,2], Chen Xiaoyi[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, jxiang,}@jaist.ac.jp,bjorner@gmail.com

June 14, 2006. Compiled January 4, 2008

Abstract

Classical digital rights license languages applied to the electronic "downloading", payment and rendering (playing) of artistic works (for example music, literature readings and movies). In this paper we generalise such applications languages and we extend the concept of licensing to also cover work authorisation (work commitment and promises) in health care and in public government. The digital works for these two new application domains are patient medical records and public government documents.

Digital rights licensing for artistic works seeks to safeguard against piracy and to ensure proper payments for the rights to render these works. Health care and public government license languages seek to ensure transparent and professional (accurate and timely) health care, respectively 'good governance'. Proper mathematical definition of licensing languages seeks to ensure smooth and correct computerised management of licenses.

In this paper we shall motivate and exemplify three license languages, their pragmatics, syntax and informal as well as formal semantics.

[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, observations, etc.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

Contents

1 Introduction **5**
1.1 What Kind of Science Is This? . . . 5
1.2 What Kind of Contributions? . . . 5

2 Pragmatics of The Three License Languages **6**
2.1 The Performing Arts: Producers and Consumers . . . 6
2.1.1 Operations on Digital Works . . . 6
2.1.2 License Agreement and Obligation . . . 6
2.1.3 The Artistic Electronic Works: Two Assumptions . . . 6
2.1.4 Protection of the Artistic Electronic Works . . . 7
2.1.5 An Artistic Digital Rights License Language . . . 7
2.2 Hospital Health Care: Patients and Medical Staff . . . 7
2.2.1 Hospital Health Care: Patients and Patient Medical Records . . . 7
2.2.2 Hospital Health Care: Medical Staff . . . 7
2.2.3 Professional Health Care . . . 7
2.2.4 A Hospital Health Care License Language . . . 7
2.3 Public Government and the Citizens . . . 8
2.3.1 The Three Branches of Government . . . 8
2.3.2 Documents . . . 8
2.3.3 Document Attributes . . . 8
2.3.4 Actor Attributes and Licenses . . . 8
2.3.5 Document Tracing . . . 8
2.3.6 A Public Administration Document License Language . . . 8

3 The Semantic Intents of Licensor and Licensee Actions **9**
3.1 Overview . . . 9
3.2 Licenses and Actions . . . 9
3.2.1 Licenses . . . 9
3.2.2 Actions . . . 9
3.2.3 Two Languages . . . 9
3.3 Sub-licensing, Scheme I . . . 9
3.4 Sub-licensing, Scheme II . . . 10
3.5 Multiple Licenses . . . 11

4 Syntax and Informal Semantics **12**
4.1 General Artistic License Language . . . 12
4.1.1 The Syntax . . . 12
4.1.2 Licenses . . . 12
Syntax Annotations: . . . 12
0: Syntax Sorts: . . . 13
1–2: License Name and License Body: . . . 13
3: Commands: . . . 13
4–6: Render, Copy and Edit: . . . 13
7: Read Me: . . . 13
8: Sub-Licensing: . . . 13
Informal Semantics: . . . 14
4.1.3 Actions . . . 14
Annotations and Informal Semantics:: . . . 14
5: Variables: . . . 14
6: Actions: . . . 14
7: Render: . . . 15
8: Copy: . . . 15
9: Edit: . . . 15
9: Edit: . . . 15
10: Sub-Licensing: . . . 15
4.2 Hospital Health Care License Language . . . 15

4.2.1 A Notion of License Execution State . . . 16
4.2.2 The Syntax . . . 16
Licenses: . . . 16
Syntax: Annotations: . . . 17
Actions: . . . 17
Syntax Annotations: . . . 17
Informal Semantics: . . . 18
4.3 Public Administration License Language . . . 18
4.3.1 The Syntax: Licenses . . . 18
The Form of Licenses: . . . 18
Licensed Operations: . . . 18
Syntax & Informal Semantics Annotations: Licenses: . . . 19
4.3.2 The Syntax: Actions . . . 20
Preliminary Remarks: . . . 21
Syntax & Informal Semantics Annotations: Actions: . . . 21
4.4 Discussion . . . 22
4.4.1 Comparisons . . . 22
Differences: . . . 22
Work Items: . . . 22
Operations: . . . 22
Permissions and Obligations: . . . 22

5 Formal Semantics **23**
5.1 A Model of Common Aspects . . . 23
5.1.1 Actors: Behaviours and Processes . . . 23
System States: . . . 23
System Processes: . . . 23
Actor Processes: . . . 24
The Pro-active Actor Behaviour: . . . 24
The Re-active Actor Behaviour: . . . 24
5.1.2 Functions . . . 25
Auxiliary Function Signatures: . . . 25
Elaboration Function Signature: . . . 25
Discussion of Auxiliary Functions: . . . 26
Schema Definitions of Elaboration Functions: . . . 26
5.2 A Model of The General Artistic License Language . . . 26
5.2.1 A Licensor/Licensee State . . . 26
5.2.2 Auxiliary Functions . . . 26
5.2.3 Elaboration Functions . . . 26
5.3 A Model of The Hospital Health Care License Language . . . 26
5.3.1 A Patient [Medical Record] State . . . 26
5.3.2 A Medical Staff State . . . 26
5.3.3 Auxiliary Functions . . . 26
5.3.4 Elaboration Functions . . . 26
5.4 A Model of The Public Administration License Language . . . 26
5.4.1 A Public Administrator State . . . 26
5.4.2 Auxiliary Functions . . . 26
5.4.3 Elaboration Functions . . . 26
5.5 Discussion . . . 26

6 Conclusion **27**
6.1 Summary: What Have We Achieved? . . . 27
6.2 Open Issues . . . 27
6.2.1 An Open Issue: Rôle of Modal Logics . . . 27
6.2.2 Another Open Issue: Fair Use . . . 27
6.3 Acknowledgements . . . 27
6.4 Bibliographical Notes . . . 28

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

A The Gunter et al. Model & its Reformulation **33**
A.1 Digital Rights Licensing . . . 33
A.1.1 What is Digital Rights? . . . 33
A.1.2 Realities by Users and Licenses Issued by Owners . . . 33
A.1.3 Digital Rights Management (DRM) . . . 33
A.1.4 Structure of Presentation . . . 33
A.2 Transcript of the Gunter/Weeks/Wright Paper . . . 33
A.2.1 Actions, Events, Realities and Licenses . . . 33
A.3 Standard Licenses . . . 36
A.3.1 Up Front Licenses . . . 36
A.3.2 Flat Rate Licenses . . . 37
A.3.3 Per Use Licenses . . . 37
A.3.4 Up to Expiry Date Licenses . . . 38
A.3.5 Non-cancellable Multi-use Licenses . . . 38
A.3.6 Cancellable Multi-use Licenses . . . 39
A.4 A License Language . . . 40
A.4.1 Syntax . . . 40
A.4.2 Examples . . . 40
A.4.3 Semantics . . . 40
A.5 An RSL Model . . . 40
A.5.1 Actions, Events, Realities and Licences . . . 40
A.6 Standard Licences . . . 42
A.6.1 Syntax . . . 42
A.6.2 Semantics . . . 43
A.7 A License Language . . . 45
A.8 End of "Gunter" Paper . . . 46

1 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 [52]

The concepts of licenses and licensing express relations between *actors* (licensors (the authority) and licensees), *entities* (artistic works, hospital patients, public administration and citizen 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. In this paper 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 related to public government.

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 (rendering, copying, editing, sub-licensing) 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 prefectures 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, perhaps copy) other documents.

1.1 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.

1.2 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, we hope.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

2 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.*

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 hospitalisation of patient, i.e., hospital health care 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.

2.1 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 paper we shall not touch upon the technical issues of "downloading"(whether "streaming" or copying, or other). That and other issues will be analysed in [55].

2.1.1 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, concert, 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. 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.

2.1.2 License Agreement and Obligation

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

2.1.3 The Artistic Electronic Works: Two Assumptions

Two, related assumptions underlie 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". In either case we "derive" the second assumption (from the fulfilment 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

[2] render, copy and edit

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 sub-licensing.

2.1.4 Protection of the Artistic Electronic Works

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

2.1.5 An Artistic Digital Rights License Language

In Sects. 4.1, 5.1, and 5.2 we shall design a suitably flexible digital artistic works license language and, through its precise informal and formal description provide one set answers to the above issue.

2.2 Hospital Health Care: 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 treated with the aim of full or partial restitution. 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.

2.2.1 Hospital Health Care: 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.

2.2.2 Hospital Health Care: 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 'referrals') in the form of hospitalisation (not treatment) plans. We shall call such scripts for licenses.

2.2.3 Professional Health Care

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.

2.2.4 A Hospital Health Care License Language

In Sects. 4.2, 5.1, and 5.3 we shall design a suitably flexible hospital health care license language and, through its precise informal and formal description provide one set answers to the above issue.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

2.3 Public Government and the Citizens

2.3.1 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. Typically national parliament and local (province and city) councils are part of law-making government, law-enforcing government is called the executive (the administration), and law-interpreting government is called the judiciary [system] (including lawyers etc.).

2.3.2 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.)

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

2.3.3 Document Attributes

With documents one can associate, as attributes of documents, the *actors* who created, edited, read, copied, distributed (and to whom distributed),shared, performed calculations and shredded documents.

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

2.3.4 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*.

2.3.5 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.

2.3.6 A Public Administration Document License Language

Sects. 4.3, 5.1, and 5.4 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.

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

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

3 The Semantic Intents of Licensor and Licensee Actions

3.1 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 item:* the artistic 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.

3.2 Licenses and Actions

3.2.1 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

3.2.2 Actions

And, conceptually, the licensee (u) may perform actions which can be expressed as follows:

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}, mentioned in the license, or they may not be in that set. In the latter case we have a breach of license ℓ.

3.2.3 Two Languages

Thus there is *the language of licenses* and *the language of actions*.

We advise the reader to take note of the distinction between the permitted or obligated actions enumerated in a license and the license name labelled actions actually requested by a licensee.

3.3 Sub-licensing, Scheme I

A licensee u may wish to delegate some of the work associated with performing some licensed actions to a colleague (or customer). 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

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

{ai,aj,...,ak} ⊆ {a1,a2,...,an}

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 sub-licensing of actions {ai,aj,...,ak}

where

{ai,aj,...,ak} ⊆ {a1,a2,...,an}

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.

3.4 Sub-licensing, Scheme II

The subset relation

{ai,aj,...,ak} ⊆ {a1,a2,...,an}

mentioned in the sub-licensing part of license

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

may be omitted. In fact one could relax the relation completely and allow any actions {ai,aj,...,ak} whether in {a1,a2,...,an} or not ! That is, the original licensor may mandate that a licensee allow a sub-licensee to perform operations that the licensee is not allowed to perform. Examples are: a licensee may break the shrink-wrap around some licensed software package — an action which may not be performed by the licensor; a medical nurse (i.e., a licensee) may perform actions on patients not allowed performed by the licensor (say, a medical doctor); and a civil servant (say, an archivist) may copy, distribute or shred documents, actions that may not be allowed by the licensor (i.e., the manager of that civil servant), while that civil servant (the archivist) is not allowed to create or read documents.

3.5 Multiple Licenses

Consider the following scenario: A licensee L is performing actions $a_p, a_q, \ldots, a_r$, on work item ω, and has licensed L' to perform actions $a_i, a_j, \ldots, a_k$, also on work item ω. The action whereby L licenses L' occurs at some time. At that time L has performed none or some of the actions $a_p, a_q, \ldots, a_r$ (on work item ω), but maybe not all. What is going to happen? Can L and L' go on, in parallel, performing actions on the same work item (ω) ? 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.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

4 Syntax and Informal Semantics

We distinguish between the pragmatics, the semantics and the syntax of languages. Leading textbooks on (formal) semantics of programming languages are [18, 28, 47, 50, 53, 54].

We have already covered the concept of pragmatics and illustrated its application to some issues of license language design.

We shall now illustrate the use of syntax and semantics in license language design.

- *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 [17].*
- By **informal** *semantics we mean a semantics which is expressed in concise natural language, for example, as here, English.*

4.1 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 syntax has been facilitated by the work reported by Xiang JenWen [55].[5]

4.1.1 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.

4.1.2 Licenses

We first present an abstract syntax of the language of licenses, then we annotate this abstract syntax, and finally we present an informal semantics of that language of licenses.

type
0. Ln, Nm, W, S, V
1. L = Ln × Lic
2. Lic == mkLic(licensor:Nm,licensee:Nm,work:W,cmds:Cmd-**set**)
3. Cmd == Rndr | Copy | Edit | RdMe | SuLi
4. Rndr = mkRndr(vw:(V|″`work`″),sl:S*)
5. Copy = mkCopy(fvw:(V|″`work`″),sl:S*,tv:V)
6. Edit = mkEdit(fvw:(V|″`work`″),sl:S*,tv:V)
7. RdMe = "readme"
8. SuLi = mkSuLi(cs:Cmd-**set**,work:V)

Syntax Annotations:

[5] As this work, [55], has yet to be completed the syntax and annotations given here may change.

0: Syntax Sorts: (0.) Licenses are given names, ln:Ln, so are actors (owners, licensors, and users, licensees), nn:Nm. By w:W we mean a (net) reference to (a name of) the downloaded possibly segmented artistic work being licensed, where segments are named (s:S), that is, s:S is a selector to either a segment of a downloaded work or to a segment of a copied and or and edited work.

1–2: License Name and License Body: (1.) Every license (lic:Lic) has a unique name (ln:Ln).

(2.) A license (lic:Lic) contains four 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).

3: Commands: (3.) A command is either a render, a copy or an edit or a readme command, or a sub-licensing (sub-license) command.

4–6: Render, Copy and Edit: (4.–6.) The render, copy and edit commands are each "decorated" with an ordered list of selectors (i.e., selector names) and a (work) variable name. The license command

copy ⟨s1,s2,s7⟩ v

means that the licensed work, ω, may have its sections s_1, s_2 and s_7 copied, in that sequence, into a new variable named v, Other copy commands may specify other sequences. Similarly for render and edit commands.

7: Read Me: (7.) The "readme" license command, in a license, ln, referring, by means of w, to work ω, somehow displays a graphical/textual "image" of, that is, information about ω. We do not here bother to detail what kind of information may be so displayed. But you may think of the following display information names of artistic work,artists, authors, etc., names and details about licensed commands, a table of fees for performing respective licensed commands, etcetera.

8: Sub-Licensing: (8.) The license command

license cmd1,cmd2,...,cmdn **on work** v
mkSuLi({cmd1,cmd2,...,cmdn},v)

means that the licensee is allowed to act as a licensor, to name sub-licensees (that is, licensees) freely, to select only a (possibly full) subset of the sub-licensed commands (that are listed) for the sub-licensee to enjoy. The license need thus not mention the name(s) of the possible sub-licensees. But one could design a license language, i.e., modify the present one to reflect such constraints. The license also do not mention the payment fee component. As we shall see under licensor actions such a function will eventually be inserted.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

Informal Semantics: A license licenses the licensee to render, copy, edit and license (possibly the results of editing) any selection of downloaded works. 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 occurs (that can be inspected by means of the **read license** command). The user can render the downloaded work and can render copies of the work as well as edited versions of these. Edited versions are given own names. 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.

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" (see next section) functions are part of the license and the licensed work 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.

Licenses specify sets of permitted actions. Licenses do not normally mandate specific sequences of actions. 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 sub-license it. But the latter is strictly speaking not necessary.

4.1.3 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)

Annotations and Informal Semantics::

5: Variables: (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.

6: Actions: (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.)

7: Render: (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.

8: Copy: (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.

9: Edit: (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.

9: Edit: 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.

10: Sub-Licensing: (10.) The licensee can further **sub-license** 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 sub-license, 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.

The sub-license 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 sub-license, become the new licensee. The assemblage refers to the context of the action. That context knows who, the licensor, is issuing the sub-license. From the license label of the command it is known whether the sub-license actions are a subset of those for which sub-licensing has been permitted.

4.2 Hospital Health Care License Language

We refer to the abstract syntax formalised below (that is, formulas 1.–5.). The work on the specific form of the syntax has been facilitated by the work reported in [2].[6]

[6] As this work, [2], has yet to be completed the syntax and annotations given here may change.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

4.2.1 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.

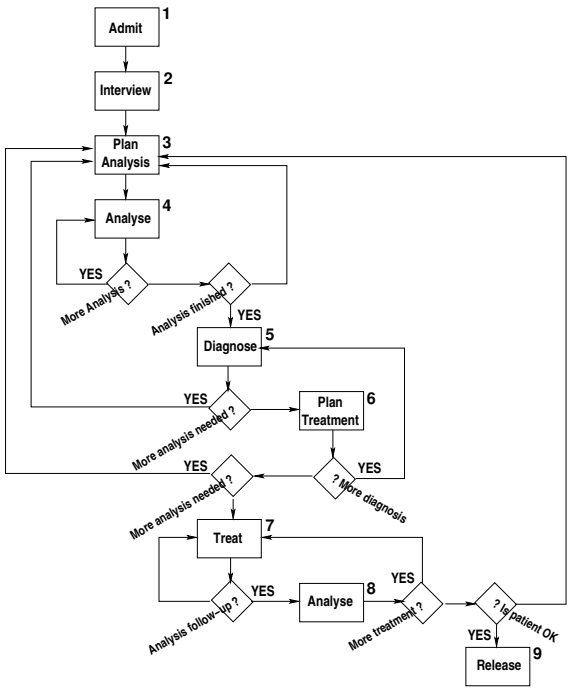

Figure 1: An example hospitalisation plan. States: {1,2,3,4,5,6,7,8,9}

We therefore suggest to join to the licensed commands an indicator which prescribe the (set of) state(s) of the hospitalisation plan in which the command action may be performed.

Two or more medical staff may now be licensed to perform different (or even same !) actions in same or different states. If licensed to perform same action(s) in same state(s) — well that may be "bad license programming" if and only if it is bad medical practice ! One cannot design a language and prevent it being misused!

4.2.2 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.

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(σs:Σ-**set**,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 [25, 26, 6, 7, 8, 24]. But it is decorated! The subtypes {|**boldface keyword**|} are inserted for readability.

Syntax: Annotations: (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 part (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 sub-license (ML) or an alternative.

(5.) A command (Cmd) is a state-labelled statement.

(3.) A sub-license just states the command set that the sub-license 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 sub-licensee 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 sub-licensee, the licensee (now new licensor) chooses to confer actions to other staff.

(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.

Actions:

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

Syntax Annotations: (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. 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.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

(10.) Or an action can be a sub-licensing action. Either the sub-licensing action that the licensee is attempting is explicitly mandated by the license (4. ML), or is an alternative one thus implicitly mandated (6.). The full sub-license, as defined in (1.–3.) is compiled from contextual information.

Informal Semantics: An informal, rough-sketch semantics (here abbreviated) would state that a prescribed action is only performed if the patient, cum patient medical record is in an appropriate state; and that the patient is being treated according to the action performed; that records of this treatment and its (partially) analysed outcome is introduced into the patient medical record. The next state of the patient, cum patient medical record, depends on the outcome of the treatment[7]; and hence the patient medical record carries with it, i.e., embodies a, or the, hospitalisation plan in effect for the patient, and a reference to the current state of the patient.

4.3 Public Administration License Language

We refer to the abstract syntax formalised below (that is, formulas 1.–15.). The work on the specific form of the syntax has been facilitated by the work reported in [5, 9, 13].[8]

4.3.1 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.

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

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

[7] Cf. the diamond-shaped decision boxes in Fig. 1 on page 16.
[8] As part this work, [13], has yet to be completed the syntax and annotations given here may change.

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)

Syntax & Informal Semantics 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, restricting 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.[9] A set of operations on (named) documents are granted.

(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).

(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[10]) 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.[11] The "based on" documents are moved from licensor to licensee.

(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 (etc.) — as per [5]. 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.

(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 (etc.) — as per [5]. 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.

(12a.) A licensee can **sub-license** (sL) certain operations to be performed by other actors.

[9] This constraint can be enforced by letting the actor name be part of the license name.

[10] — hence there is an assumption here that the create operation is invoked by the licensee exactly (or at most) once.

[11] They can therefore be traced (etc.) — as per [5].

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

(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 sub-license), and leaves it open to the licensor to freely choose a licensee. One could, instead, for example, constrain the licensor to choose from a certain class of actors. The licensor (the licensee issuing the sub-license) must choose a unique license name.

(13.) A document can be **shared** between two or more actors. One of these is the licensee, the others are implicitly given read authorisations. (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.

(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.

(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 relinquishes all control over that document.

(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.

4.3.2 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)

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".

Syntax & Informal Semantics 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.

(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.

(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, perhaps, a withdrawal.

(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.) Sharing may (conditionally) be **revoked**, partially or fully, that is, wrt. original "sharers".

(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.

(28.) A license may (conditionally) apply any of a licensed set of **calculation functions** to "downloaded" (edited, copied, etc.) documents, or can (unconditionally) 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.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

4.4 Discussion

4.4.1 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 be given after we have formalised these languages. The comparisons can be properly founded on comparing the semantic values of these languages.

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 temporary one.

Differences:

Work Items: The **work items** of the **artistic** license language(s) are essentially "kept" by the licensor. The **work items** of the **hospital health care** license language(s) are fixed and, for a large 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.

Operations: The **operations** of the **artistic** license language(s) are are essentially "kept" by the licensor. The **operations** of the **hospital health care** 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).

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 health care** 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 Formal Semantics

By formal semantics we understand a definition expressed in a formal language, that is, a language with a mathematical syntax, a mathematical semantics, and a consistent and relative complete proof system. We shall deploy the CSP [32, 48, 51, 33] Specification Language embedded in a Landin–like notation of **let** clauses[12]. We hope someone will complement our definition with a commensurate CafeOBJ [23, 22, 20, 19] definition.

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. So the system is then modelled as a set of concurrent behaviours, that is, parallel (‖) CSP processes. Actors are uniquely identified (Aid).

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

type
  Aid, Ω
  Ωs = Aid $\overrightarrow{m}$ Ω

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 (⌈⌉) alternates between these activities.

type
  M = Lic | Pay | Rpt | ...
channel
  {a[i]|i:Aid} (Aid×M)
value
  actor: j:Aid × Ω → **in**,**out** {a[i]|i:Aid•i≠j} **Unit**
  actor(j)(ω) ≡ **let** ω′ = pro_act(j)(ω)⌈⌉re_act(j)(ω) **in** actor(j)(ω′) **end**

  system: Ωs → **Unit**
  system(ωs) ≡ ‖ {actor(i)(ωs(i))|i:Aid}

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

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

Actor Processes: We have identified two kinds of actor processes: pro-active, during which the actor, by own initiative, (as a prospective 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 requester becomes a licensee), or "downloads" (access to) requested works or functions. functions.

The Pro-active Actor Behaviour: In the pro-active behaviour an actor, at will, i.e., non-deterministically internal choice (⌈⌉), decides to either request a license (rl) or to perform some action (op). In the former case the actor inquires (l_iq) an own state to find out from which licensor (aid) which kind of license requirements (l_rq) is to be requested. This licensor and these requirements are duly noted in the state. After sending the request the actor continues being an actor in the duly noted state. In the latter case (op) there may be many "next" actions to do (act). The actor inquires (a_iq) an own state to find out which action (designated by op_i) is "next". The actor them performs (act) the designated operation. It is here, for simplification assumed that all operation completions imply a "completion" message (a payment, a report, or other) to the operation licensing actor. So such a message is sent and the operation updated local state is yielded — whereby the pro-active actor "resumes" being an actor as from that state.

```
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
```

The Re-active Actor Behaviour: In the re-active behaviour an actor (j), is willing to engage in communication with other actors. This is formalised by a non-deterministic external choice ([]) between either of a set ({...}) of (zero, or more) other actors (k:Aid\{j}) who are trying to contact the re-active actor. The communicated message reveals the identity (k) of the requesting, i.e., the pro-active actor,[13] The message, m, reveals what action (act(m)) the re-active actor is requested to perform. The actor does so/ This results in a reply message m′

[13]Do not get confused by the two k's on either side of the let clause. The left k is yielded by the (input) communication a(k)?. The defining scope of the right side k, as in a(k), is just the right-hand side of the left clause.

and a state change. The reply message is sent to the requesting actor; and the re-active actor yields the requested action-updated state — whereby the re-active actor "resumes" being an actor as from that state.

```
type
  M = Lic | Pay | Rpt | ...
channel
  {a[ i ]|i:Aid} (Aid×M)
value
  re_act: j:Aid → Ω → in,out {a[ i ]|i:Aid•j≠i}  Ω
  re_act(j)(ω)≡
    let (k,m)=[]{a(k)?|k:Aid} in let (m′,ω′)=act(m)(ω) in a(k)!(j,m′);ω′ end end
```

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 processes. After that we discuss the former and suggest definitions of the latter.

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: Ω → Aid × License_Requirements × Ω
  iq_a_Ω: Ω → Action_Designator
```

By 'eureka'[14] 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.

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 which to have a payment, a report, or some such thing "back"!

```
type
  Action_Designation
value
  act: Action_Designation → Ω → Aid × M × Ω
```

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

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

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, as it were, which inquiries 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.

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 Health Care 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 achieve 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; a 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 of each individual language.

6.2 Open Issues

6.2.1 An Open Issue: Rôle of Modal Logics

Temporal logic Deontic logic (permission and obligation) Logic of knowledge and belief etc.

6.2.2 Another Open Issue: Fair Use

Fair use: ... etc.

6.3 Acknowledgements

The second author wishes to express that it has been an interesting experience to work with his three co-author JAIST "students": Mr. Yasuhito Arimoto, MSc student, Miss Chen Xiaoyi, PhD student, and Dr. Xiang Jianwen, Post-doc.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

6.4 Bibliographical Notes

We comment on some papers on the topic of digital rights:

- [3]
- [4]
- [14]
- [16]
- [15]
- [35]
- [12]
- [27]
- [29]
- [34]
- [37]
- [36]
- [38]
- [40]
- [42]
- [43]
- [1]
- [44]
- [45]
- [49]

References

[1] O.M. Alliance. OMA DRM V2.0 Candidate Enabler. http://www.openmobile-alliance.org/release_program/drm_v2_0.html, Sep 2005.

[2] Y. Arimoto, D. Bjørner: Hospital Healthcare: A Domain Analysis and a License Language. Technical Note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292 (2006)

[3] A. Arnab, A. Hutchison: Fairer Usage Contracts for DRM. In: *Proceedings of the Fifth ACM Workshop on Digital Rights Management (DRM'05)* (2005) pp 65–74

[4] Y. Benkler: *Coase's Penguin, or Linux and the Nature of the Firm.* The Yale Law Journal **112** (2002)

[5] D. Bjørner: Documents: A Domain Analysis. Technical Note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292 (2006)

[6] D. Bjørner: *Software Engineering, Vol. 1: Abstraction and Modelling* (Springer, 2006)

[7] D. Bjørner: *Software Engineering, Vol. 2: Specification of Systems and Languages* (Springer, 2006)

[8] D. Bjørner: *Software Engineering, Vol. 3: Domains, Requirements and Software Design* (Springer, 2006)

[9] D. Bjørner, X. Chen: Public Government: A Domain Analysis. Technical Note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292 (2006)

[10] Edited by D. Bjørner, M.C. Henson: *Logics of Specification Languages — see [46, 11, 19, 41, 30, 24, 39, 21, 31]* (Springer, Heidelberg, Germany 2008)

[11] D. Cansell, D. Méry. *Logics of Specification Languages*, chapter The event-B Modelling Method: Concepts and Case Studies, pages in [10], 47–152. Springer, 2008.

[12] C.E.C. Digital Rights: Background, Systems, Assessment. Commission of The European Communities, Staff Working Paper, 2002. Brussels, 14.02.2002, SEC(2002) 197.

[13] X. Chen, D. Bjørner: Public Government: A Domain Analysis and a License Language. Technical Note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292 (2006)

[14] C.N. Chong, R. Corin, S. Etalle: LicenseScript: A novel digital rights languages and its semantics. In: *Proc. of the Third International Conference WEB Delivering of Music (WEDELMUSIC'03)* (IEEE Computer Society Press, 2003) pp 122–129

[15] C.N. Chong, R.J. Corin, J.M. Doumen et al: *LicenseScript: a logical language for digital rights management.* Annals of telecommunications special issue on Information systems security (2006)

[16] C.N. Chong, S. Etalle, P.H. Hartel: Comparing Logic-based and XML-based Rights Expression Languages. In: *Confederated Int. Workshops: On The Move to Meaningful Internet Systems (OTM)*, no 2889 of *LNCS* (Springer, Catania, Sicily, Italy 2003) pp 779–792

[17] D. Crystal: *The Cambridge Encyclopedia of Language* (Cambridge University Press, 1987, 1988)

[18] J. de Bakker: *Control Flow Semantics* (The MIT Press, Cambridge, Mass., USA, 1995)

[19] R. Diaconescu. *Logics of Specification Languages*, chapter A Methodological Guide to the CafeOBJ Logic, pages in [10], 153–240. Springer, 2008.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

[20] R. Diaconescu, K. Futatsugi, K. Ogata: *CafeOBJ: Logical Foundations and Methodology.* Computing and Informatics **22**, 1–2 (2003)

[21] J.S. Fitzgerald. *Logics of Specification Languages*, chapter The Typed Logic of Partial Functions and the Vienna Development Method, pages in [10], 453–487. Springer, 2008.

[22] K. Futatsugi, R. Diaconescu: *CafeOBJ Report The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification* (World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, SINGAPORE 596224. Tel: 65-6466-5775, Fax: 65-6467-7667, E-mail: wspc@wspc.com.sg 1998)

[23] K. Futatsugi, A. Nakagawa, T. Tamai, editors. *CAFE: An Industrial–Strength Algebraic Formal Method*, Sara Burgerhartstraat 25, P.O. Box 211, NL–1000 AE Amsterdam, The Netherlands, 2000. Elsevier. Proceedings from an April 1998 Symposium, Numazu, Japan.

[24] C. George, A.E. Haxthausen. *Logics of Specification Languages*, chapter The Logic of the RAISE Specification Language, pages in [10], 349–399. Springer, 2008.

[25] C.W. George, P. Haff, K. Havelund et al: *The RAISE Specification Language* (Prentice-Hall, Hemel Hampstead, England 1992)

[26] C.W. George, A.E. Haxthausen, S. Hughes et al: *The RAISE Method* (Prentice-Hall, Hemel Hampstead, England 1995)

[27] C.A. Gunter, S.T. Weeks, A.K. Wright: Models and Languages for Digtial Rights. In: *Proc. of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)* (IEEE Computer Society Press, Maui, Hawaii, USA 2001) pp 4034–4038

[28] C. Gunther: *Semantics of Programming Languages* (The MIT Press, Cambridge, Mass., USA, 1992)

[29] J.Y. Halpern, V. Weissman: A Formal Foundation for XrML. In: *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)* (2004)

[30] M.R. Hansen. *Logics of Specification Languages*, chapter Duration Calculus, pages in [10], 299–347. Springer, 2008.

[31] M.C. Henson, M. Deutsch, S. Reeves. *Logics of Specification Languages*, chapter Z Logic and Its Applications, pages in [10], 489–596. Springer, 2008.

[32] T. Hoare: *Communicating Sequential Processes* (Prentice-Hall International, 1985)

[33] T. Hoare. Communicating Sequential Processes. Published electronically: `http://www.-usingcsp.com/cspbook.pdf`, 2004. Second edition of [32]. See also `http://www.-usingcsp.com/`.

[34] M. Holzer, S. Katzenbeisser, C. Schallhart: Towards a Formal Semantics for ODRL. In: *Proc. of the First International ODRL Workshop* (2004)

[35] C. Inc. XrML: Extensible rights Markup Language. http://www.xrml.org, 2000.

[36] R. Koenen, J. Lacy, M. Mackay, S. Mitchell: *The long march to interoperable digital rights management.* Proceedings of the IEEE **92**, 6 (2004) pp 883–897

[37] I.S.P. Ltd. Open Digital Rights Language (ODRL). http://odrl.net, 2001.

[38] G.E. Lyon: Information Technology: A Quick-Reference List of Organizations and Standards for Digital Rights Management. NIST Special Publication 500-241, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce (2002)

[39] S. Merz. *Logics of Specification Languages*, chapter The Specification Language TLA$^+$, pages in [10], 401–451. Springer, 2008.

[40] S. Michiels, K. Verslype, W. Joosen, B.D. Decker: Towards a Software Architecture for DRM. In: *Proceedings of the Fifth ACM Workshop on Digital Rights Management (DRM'05)* (2005) pp 65–74

[41] T. Mossakowski, A. Haxthausen, D. Sannella, A. Tarlecki. *Logics of Specification Languages*, chapter CASL – the Common Algebraic Specification Language, pages in [10], 241–298. Springer, 2008.

[42] D. Mulligan, A. Burstein: Implementing copyright limitations in rights expression languages. In: *Proc. of 2002 ACM Workshop on Digital Rights Management*, vol 2696 of *Lecture Notes in Computer Science* (Springer-Verlag, 2002) pp 137–154

[43] D.K. Mulligan, J. Han, A.J. Burstein: How DRM-Based Content Delivery Systems Disrupt Expectations of "Personal Use". In: *Proc. of The 3rd International Workshop on Digital Rights Management* (2003) pp 77–89

[44] R. Pucella, V. Weissman: A Logic for Reasoning about Digital Rights. In: *Proc. of the 15th IEEE Computer Security Foundations Workshop (CSFW'02)* (IEEE Computer Society Press, 2002) pp 282–294

[45] R. Pucella, V. Weissman: A Formal Foundation for ODRL. In: *Proc. of the Workshop on Issues in the Theory of Security (WIST'04)* (2004)

[46] W. Reisig. *Logics of Specification Languages*, chapter Abstract State Machines for the Classroom, pages in [10], 15–46. Springer, 2008.

[47] J.C. Reynolds: *The Semantics of Programming Languages* (Cambridge University Press, 1999)

[48] A.W. Roscoe: *Theory and Practice of Concurrency* (Prentice-Hall, 1997)

[49] P. Samuelson: *Digital rights management {and, or, vs.} the law.* Communications of ACM **46**, 4 (2003) pp 41–45

[50] D.A. Schmidt: *Denotational Semantics: a Methodology for Language Development* (Allyn & Bacon, 1986)

[51] S. Schneider: *Concurrent and Real-time Systems — The CSP Approach* (John Wiley & Sons, Ltd., Baffins Lane, Chichester, West Sussex PO19 1UD, England 2000)

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

[52] Staff of Merriam Webster. Online Dictionary: `http://www.m-w.com/home.htm`, 2004. Merriam–Webster, Inc., 47 Federal Street, P.O. Box 281, Springfield, MA 01102, USA.

[53] R. Tennent: *The Semantics of Programming Languages* (Prentice–Hall Intl., 1997)

[54] G. Winskel: *The Formal Semantics of Programming Languages* (The MIT Press, Cambridge, Mass., USA, 1993)

[55] J. Xiang, D. Bjørner: The Electronic Media Industry: A Domain Analysis and a License Language. Technical Note, JAIST, School of Information Science, 1-1, Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan 923-1292 (2006)

A The Gunter et al. Model & its Reformulation

A.1 Digital Rights Licensing

A.1.1 What is Digital Rights?

- Digital rights deal with the rights of owners of artistic expressions
  - music, movies, readings, photographs, paintings, ...
- in the context of the downloading of these
- via the Internet to users
- who are then supposed to pay for the right to render,
- i.e., to listen, see, hear, and see–see, ..., these.

A.1.2 Realities by Users and Licenses Issued by Owners

- Users perform **pay**ment and **render**ing **events**.
- Sequences of events as performed by users make up **realities**.
- Owners issue **licenses** which describe which realities are permissible.

A.1.3 Digital Rights Management (DRM)

- DRM is then about
  - the **design** of appropriate **license languages**
  - and the **enforcement** of user realities
  - in order for these to **not breach**, but to **fullfil** the licenses.

A.1.4 Structure of Presentation

- First a loyal, but problematic transcription of a published paper.
- Then a "similar", but believed correct reformulation (in RSL).

A.2 Transcript of the Gunter/Weeks/Wright Paper

A.2.1 Actions, Events, Realities and Licenses

- The model centers around
  - a domain of *realities* and
  - a domain of *licenses*,
- where
  - A *reality* is a sequence of events.

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

  - A *license* is a set of realities.

- The semantics of a rights management languages can be expressed as a function
- that maps terms of the language to elements of their domain of licenses.
- Their abstract model
  - represents an *event*, $e \in Event$, as a pair of a *time*, $t \in Time$,
  - and an *action*, $a \in Action$:

$$e ::= t : a$$

- *Time* is totally ordered by $<$.
- The function $+$:
  - $Time \times Period \rightarrow Time$
  - adds a period, $p \in Period$, to a time.
- There are two kinds of actions:

$$a ::= \mathsf{render}[w, d] \mid \mathsf{pay}[x]$$

- $w \in Work$ denotes a rights-managed work.
- $d \in Device$ represents a DRM-enabled device.
- x is a decimal.
- Action $\mathsf{render}[w, d]$ represents rendering of work w by rights-enabled device d.
- Action $\mathsf{pay}[x]$ represents a payment of amount x of some currency from a licensee to a license issuer.
- The event $t : \mathsf{render}[w, d]$ means that at time t, work w was rendered on device d.
- The event $t : \mathsf{pay}[x]$ means that at time t, a payment of amount x was made.
- A *reality*, $r \in Reality$,
  - is a finite set of events,
  - such that all events occur at distinct times:

$$\text{Reality} = \{ \text{ E} \mid \text{E:Event-}\textbf{set} \bullet \text{t:a} \in \text{E} \wedge \text{t:a}' \in \text{E} \Rightarrow \text{a=a}' \}$$

  where

  - $P(x)$ represents the powerset of E (the set of all subsets of E).
  - Notation:
    - $r_{\leq t}$ represents the prefix of r that occurs at or before time t;

    * that is: $r_{\leq t} = \{t' : a \in r \mid t' \leq t\}$.
  - Notation:
    * $r \sqsubseteq r'$ indicates that r is a prefix of r';
    * that is, there exists a t such that $r = r'_{\leq t}$.

- In the model, a *license*, $l \in License$, is a set of realities:

$$l \in License = P(Reality)$$

- Let us take an example:

$$l_A = \left\{ \begin{array}{l} \{8:00 : \mathsf{pay}[\$1],\ 8:01 : \mathsf{render}[w_1, d_1]\}, \\ \{8:00 : \mathsf{pay}[\$1],\ 8:02 : \mathsf{render}[w_1, d_1]\}, \\ \{8:00 : \mathsf{pay}[\$1],\ 8:03 : \mathsf{render}[w_1, d_1], \\ \qquad\qquad\qquad 8:04 : \mathsf{render}[w_1, d_1]\}, \\ \{8:00 : \mathsf{pay}[\$1]\} \end{array} \right\}$$

- To give a more formal meaning to a license,
- suppose **r** is the (unique) complete reality
- that actually occurs over the entire lifetime of the DRM system.
- Let $\mathbf{r}[l]$ be events of **r** attributed to license l.
- **Definitions**
  - Reality $r \in l$ of license l is *viable* for $\mathbf{r}[l]$ at t iff $\mathbf{r}[l]_{\leq t} \sqsubseteq r$.
  - License l is *fulfilled* by $\mathbf{r}[l]$ at t iff $\mathbf{r}[l]_{\leq t} \in l$.
  - License l is *breached* by $\mathbf{r}[l]$ at t iff there does not exist $r \in l$ that is viable for $\mathbf{r}[l]$ at t.

- **Example 1:**

  $\mathbf{r}[l_A]$ = { 8:00:pay[$1], 8:01:render$[w_1, d_1]$, 8:05:render$[w_1, d_1]$ }.

  For $\mathbf{r}[l_A]$,
  - at $t <$ 8:01, all four realities of license l_A are viable
  - at t, 8:01 $\leq t <$ 8:05, only the first reality is viable
  - at $t \geq$ 8:05, no reality is viable

- **Example 2:**

  $\mathbf{r}[l_A]$ = { 8:00:pay[$1], 8:01:render$[w_1, d_1]$, 8:05:render$[w_1, d_1]$ }.

  License l_A is
  - unfulfilled by $\mathbf{r}[l_A]$ for $t <$ 8:00

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

  - fulfilled for 8:00 $\leq t <$ 8:05
  - breached for $t \geq$ 8:05

- **Example 3:**
  r'$[l_A]$ = { 8:00:pay[\$1], 8:03:render$[w_1, d_1]$ }.

  license l_A is

  - unfulfilled for $t <$ 8:00
  - fulfilled for 8:03 $\leq t <$ 8:03
  - unfulfilled for 8:03 $\leq t <$ 8:04
  - breached for $t \geq$ 8:04

A.3 Standard Licenses

A.3.1 Up Front Licenses

- The "UP Front" license provides access
- to any work in set $W \in P(Work)$
- on any device in set $D \in P(Device)$
- beginning at time t_0 for period p,
- for an up-front payment of x:

$$
\begin{aligned}
&\text{UpFront}(t_0, x, p, W, D) = \\
&\quad \{t_0 : \mathsf{pay}[x], \\
&\qquad t_1 : \mathsf{render}[w_1, d_1], ..., t_n : \mathsf{render}[w_n, d_n] \\
&\quad | \; n \geq 0, \\
&\qquad t_0 < t_1 < ... < t_n < t_0 + p, \\
&\qquad w_1, ..., w_n \in W, d_1, ..., d_n \in D \; \}
\end{aligned}
$$

Problems

- The use of n is confusing.
- On one hand it is used to express up to n renderings.
- On the other hand subscript $_n$ is used for ranging both works and devices.
- The three uses of n should be separated into, say, i, j and k.
- The same comments apply to the Flat Rate and Per Use formulations.
- Why not allow $t_n \leq t_0 + p$?

A.3.2 Flat Rate Licenses

- The "Flat Rate" license provides access
- to any work in set W
- on any device in set D
- beginning at time t_0 for period p,
- for a payment of x at the end of the period:

$$
\begin{aligned}
&\text{FlatRate}(t_0, x, p, W, D) = \\
&\quad \{t_1 : \mathsf{render}[w_1, d_1], ..., t_n : \mathsf{render}[w_n, d_n] \\
&\quad\quad t_{n+1} : \mathsf{pay}[x], \\
&\quad | \; n \geq 0, \\
&\quad\quad t_0 < t_1 < ... < t_n < t_{n+1} < t_0 + p, \\
&\quad\quad w_1, ..., w_n \in W, d_1, ..., d_n \in D \; \}
\end{aligned}
$$

Problems

- Why not allow $t_n \leq t_0 + p$?

A.3.3 Per Use Licenses

- The "Per Use" license is provides access
- to any work in set W
- on any device in set D
- beginning at time t_0
- for a period of length p,
- for a payment of x per use at the end of the period:

$$
\begin{aligned}
&\text{PerUse}(t_0, x, p, W, D) = \\
&\quad \{t_1 : \mathsf{render}[w_1, d_1], ..., t_n : \mathsf{render}[w_n, d_n] \\
&\quad\quad t_{n+1} : \mathsf{pay}[nx], \\
&\quad | \; n \geq 0, \\
&\quad\quad t_0 < t_1 < ... < t_n < t_{n+1} < t_0 + p, \\
&\quad\quad w_1, ..., w_n \in W, d_1, ..., d_n \in D \; \}
\end{aligned}
$$

Problems

- Why not allow $t_{n+1} \leq t_0 + p$?

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

A.3.4 Up to Expiry Date Licenses

- A license that a consumer can accept any time before some future
  - can be constructed by quantifying over t_0
  - for any of the three families defined on preceding slides.
- For example,

$$\bigcup_{t_0<t_{\text{expires}}} \text{UpFront}(t_0, x, p, W, D)$$

- This license allows the period of the use to begin anytime prior to t_{expires}.

Problems

- Why not allow $t_0 \leq t_{\text{expires}}$?

A.3.5 Non-cancellable Multi-use Licenses

- To construct multi-period non-cancellable licenses
- by composing single-period licenses,
- an operator $\triangle$ is defined.

$$l_1 \triangle l_2 = \{r_1 \cup r_2 \mid r_1 \in l_1, r_2 \in l_2\}$$

- here all of the events of l_1 occur at different times from the events of l_2.

Problems

- The above is not the same as: $l_1 \triangle l_2 = l_1 \cup l_2$
- How does $\triangle$ associate?
- We guess: $l \triangle\ l' \triangle\ l'' = l \triangle\ (l' \triangle\ l'')$

- $\text{UpFront}^{\triangle}(t_0, x, p, W, D, m)$
- provides access to any work in set W
- on any device in set D
- beginning at time t_0
- for m periods of length p,

- for payments of x at the beginning of each period:

$$\begin{aligned}&\text{UpFront}^{\triangle}(t_0, x, p, W, D, m) = \\ &\quad \text{UpFront}(t_0, x, p, W, D)\triangle ...\triangle \\ &\quad \text{UpFront}(t_{m-1}, x, p, W, D) \\ &\quad\quad \text{where } t_i = t_0 + ip \text{ for } i \text{ from } 0 \text{ to } m-1\end{aligned}$$

A.3.6 Cancellable Multi-use Licenses

- With the $\triangleright$ operator cancelable multi-period licenses can be defined.

$$l_1 \triangleright l_2 = \{r_1 \cup r_2 \mid r_1 \in l_1,\ r_1 \neq \emptyset,\ r_2 \in l_2\} \cup \{\emptyset\}$$

- here all of the events of l_1 occur at different times from the events of l_2.

Problems

- Same question concerning associativity.
- What, exactly is the rôle of the empty set {}, or, rather, {{}} (in Gunter et al. paper: {Ø}).
- We guess: To have only actions from l_1 and then "get out"!
- Also: There is a probem with t_i: it is defined but never used!

- $\text{UpFront}^{\triangleright}(t_0, x, p, W, D, m)$
- provides access to any work in set W
- on any device in set D
- beginning at time t_0
- for m periods of length p,
- for payments of x at the beginning of each period,
- cancelable after any period:

$$\begin{aligned}&\text{UpFront}^{\triangleright}(t_0, x, p, W, D, m) = \\ &\quad \text{UpFront}(t_0, x, p, W, D) \triangleright ...\triangleright \\ &\quad \text{UpFront}(t_{m-1}, x, p, W, D) \\ &\quad\quad \text{where } t_i = t_0 + ip \text{ for } i \text{ from } 0 \text{ to } m-1\end{aligned}$$

Problems

- Same question concerning associativity.
- There is still a probem with t_i: it is defined but never used!

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

A.4 A License Language

A.4.1 Syntax

The language *DigitalRights* is defined by the following grammar:

e ::= (at t | until t)
(for p | for [up to] m p)
pay x (upfront | flatrate | peruse)
for W on D

A.4.2 Examples

until 01/01/03
for up to 12 months
pay $ 10.00 upfront
for "Jazz Classics"
on "devices registered to license holeder"

A.4.3 Semantics

$$\mathcal{M}[\text{at } t\ z] = \mathcal{M}_1[z](t)$$
$$\mathcal{M}[\text{until } t\ z] = \bigcup_{t'<t} \mathcal{M}_1[z](t')$$

$$\mathcal{M}_1[\text{for } p\ z](t_0,p) = \mathcal{M}_2[z](t_0,p)$$
$$\mathcal{M}_1[\text{for up to } m\ p\ z](t_0) = \mathcal{M}_2[z](t_0,p) \rhd ... \rhd \mathcal{M}_2[z](t_{m-1},p)$$
$$\text{where } t_i = t_0 + ip \text{ for } i \text{ from } 0 \text{ to } m-1$$
$$\mathcal{M}_1[\text{for } m\ p\ z](t_0) = \mathcal{M}_2[z](t_0,p) \triangle ... \triangle \mathcal{M}_2[z](t_{m-1},p)$$
$$\text{where } t_i = t_0 + ip \text{ for } i \text{ from } 0 \text{ to } m-1$$

$$\mathcal{M}_2[\text{pay } x \text{ upfront for } W \text{ on } D](t,p) = \text{UpFront}(t,x,p,W,D)$$
$$\mathcal{M}_2[\text{pay } x \text{ flatrate for } W \text{ on } D](t,p) = \text{FlatRate}(t,x,p,W,D)$$
$$\mathcal{M}_2[\text{pay } x \text{ peruse for } W \text{ on } D](t,p) = \text{PerUse}(t,x,p,W,D)$$

Problems

- Parameters (t_0,p) in first lefthand side of $\mathcal{M}_1$ is wrong,
- should be just (t_0).

A.5 An RSL Model

A.5.1 Actions, Events, Realities and Licences

```
type
  T, W, D, P
  E = T × A
  A == mkR(w:W,d:D) | mkP(x:P) or: rndr(w,d) | pay(x)
  R = {| evs:E-set • wfEvs(evs) |}
```

Annotations

- T, W, D, and P stands for time, work, device and payment.
- Events, e:E, are Cartesian pairs of times (i.e., time stamps) and actions.
- Actions are discrimated, either renderings (which are records mkR(w,d)) of works and devices. or payments (mkP(x)) of currency amounts.
- The trailing "or" shows our schematic way of representing actions.
- Realities, r:R, are well-formed sets of events.

value
wfEvs: E-**set** → **Bool**
wfEvs(evs) ≡ ∀ (t,a),(t′,a′):E•{(t,a),(t′,a′)}⊆evs ∧ t=t′ ⇒ a=a′

$r_{\leq t}$: prefix: R → T → R
$r_{\leq t}$ ≡ prefix(r)(t) ≡ {(t′,a)|(t′,a):E • (t′,a) ∈ r ∧ t′≤t}

$r' \sqsubseteq r$: is_prefix: R × R → **Bool**
$r' \sqsubseteq r$ ≡ is_prefix(r′,r) ≡ ∃ t:T • r′ = prefix(r)(t)

Annotations

- A set of events form a reality if no two events have the same time stamp.
- The prefix of a reality up to and including time t is the set of all those events of the reality whose time stamp is equal to or less than t.
- A reality is a prefix of another event if there is a time t such that the former reality is a prefix of the latter reality.

type
L = R-**set**

value
/∗ viable: capable of working ∗/
is_viable: R → R → L → T → **Bool**
is_viable(r′)(r)(l)(t) ≡ r ∈ l ∧ is_prefix(prefix(r′)(t),r)

/∗ fulfilled: a consumer reality r′ satisfies a license reality ∗/
is_fulfilled: R → L → T → **Bool**
is_fulfilled(r′)(l)(t) ≡ prefix(r′)(t) ∈ l

is_breached: R → L → T → **Bool**
is_breached(r′)(l)(t) ≡ ∼∃ r:R • r ∈ l ∧ is_viable(r′)(r)(l)(t)

Annotations

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

- A reality r' is viable at a time t wrt. a reality r of a license l if r' is a prefix, at that time, of r.
- A reality r' is, at time t fulfilled wrt. a license l if the prefix of r' at time t is a reality of the license.
- License l is breached by consumer reality r' if there is no reality r in l that is viable for r' at t.

A.6 Standard Licences

A.6.1 Syntax

type
  I
  Std_L = UpFront|FlatRate|PerUse|Until|NonCanMuUpFro|CanMuUpFro
  Basics = T×P×I×W-**set**×D-**set**
  UpFront == mkUF(sb:Basics)
  FlatRate == mkFR(sb:Basics)
  PerUse == mkPU(sb:Basics)
  Until == mkU(sb:Basics,st:T)
  NonCanMuUpFro == mkNCMF(sb:Basics,sm:**Nat**)
  CanMuUpFro == mkCMF(sb:Basics,sm:**Nat**)

Annotations

- I stands for an interval, i,e,, a time period.
- Std_L stands for standard licenses.
- There are six forms of standard licences: UpFront, FlatRate, PerUse, Until, NonCanMuUpFro and CanMuUpFro.
- They all share some basic information Basics, a Time limit, Payment amount, Interval, identification of the set of Works covered by the license, and identification of the set of Devises overed by the license.
- The NonCanMuUpFro and CanMuUpFro commands furth specify a natural, usually non-zero number (of times of rendering).
- The Std_ definition defines the set of commands as a union using the | type constructor.
- Each individual type is then defined by distinctly named record type constructors: mkUF, mkFR, mkPU, mkU, mkNCMF and mkCMF.
- We have for ease of recalling these mnemonics chosen to name the conctructors with an initial mk (for 'make') and then an abbreviation of the type name being defined.
- The s...: parts of the body of the record type expressions designate selector functions.
- Meta-linguistically:

```
type
  A, B, ..., C
  R == mkR(sa:A,sb:B,...,sc:C)
axiom
  ∀ r:R, a:A, b:B, ..., c:C •
    r = mkR(sa(r),sb(r),...,sc(r)) ∧
    a = sa(mkR(a,b,...,c)) ∧
    b = sb(mkR(a,b,...,c)) ∧
    ... ∧
    c = sc(mkR(a,b,...,c))
```

A.6.2 Semantics

```
value
  M: Std_L → L
  M(mkUF(t0,x,p,ws,ds)) ≡
    {{(t0,pay(x))} ∪
      let ls = [ ti↦rndr(w,d)|ti:T,w:W,d:D•ti ∈ ts∧w ∈ ws′∧d ∈ ds′ ] in
      {(ti,ls(ti))|ti:T•ti ∈ dom ls} end
    |n:Nat,ts:T-set,ws′:W-set,ds′:D-set•
    card ts=n∧ws′⊆ws∧ds′⊆ds∧t0<min(ts)∧max(ts)≤t0+p}
```

Annotations

- The above formulation follows that of Gunter et al.,
- but where their model is plain wrong: it does not designate all combinations of works and devices, ours is right:
  - At t_0 payment is issued;
  - the above expression[15] has two set comprehensions:
  - {{A} ∪ {B|C•D} | E•F }
  - The inner comprehension {B|C•D} expresses all possible sequences of n renderings involving any combination of works w and devices d from subsets ws and ds of works anbd devices.
  - The outer comprehension "selects" an arbitrary, finite n, a set ts of n time points all of which lies between t_0 and $t_0 + p$, and arbitrary subsets ws and ds or works and devices of those given,
  - The inner comprehension ensures that all we express all runs of renderings of length n over all combinations of works and devices.
  - The outer comprehension ensures that we express all possible and indefinite length runs.

[15] { {(t0,pay(x))} ∪ {(ti,rndr(w,d)) | ti:T,w:W,d:D • ti∈ts ∧ w∈ws′ ∧ d∈ds′ } | n:Nat,ts:T-set,ws′:W-set,ds′:D-set • cardts=n ∧ ws′⊆ws ∧ ds′⊆ds ∧t 0<min(ts) ∧ max(ts)≤t0+p }

© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

- A better model would be one which, instead of constructively designating all possible runs, expresses the property that a run is an up front run and all such, for the given arguments, are present.

```
type
  PayEvent = T × ({|pay|}×Nat)
  RndEvent = T × ({|render|}×W×D)
  UpFrontLicense = {| ℓs:L • wf_UPL(ℓs) |}
value
  wf_UPL: L → Bool
  wf_UPL(ℓ) ≡ ∃ t:T,x:Nat,p:P,ws:W-set,ds:D-set • is_ufl(t0,x,p)(ws,ds,ts)(ℓ)

  is_ufl: (T × Nat × P) → (W-set × D-set × T-set) → L → Bool
  is_ufl(t0,x,p)(ws,ds,ts)(ℓ) ≡
    ∃ t′:T,x′:X • (t′,(pay,x′)) ∈ ℓ⇒ t′=t0 ∧ x′=x ∧
    ∀ ti:T,w:W,d:D • ti ∈ ts ∧ w ∈ ws ∧ d ∈ ds ⇒
        (ti,(render,w,d)) ∈ ℓ∧
    ∼∃ (t,(render,w′,d′)):RndEvent •
        (t,(render,w′,d′)) ∈ ℓ∧ t∉ ts ∧ w′∉ ws ∧ d′∉ ds
```

- A set ℓs of licenses
- is the set of all up front licenses
- designated by M(mkUF(t0,x,p,ws,ds))
- if it satisfies are_all_ufl(t0,x,p)(ws,ds,ts)(ℓs).

```
value
  are_all_ufl: T × Nat × P × W-set × D-set → L-set → Bool
  are_all_ufl(t0,x,p)(ws,ds,ts)(ℓs) ≡
    ∀ ℓ:L • ℓ∈ ℓs ⇒
      ∃ n:Nat,ws′:W-set,ds′:D-set,ts:T-set •
        ws′⊆ws ∧ ds′⊆ds ∧ card ts=n ⇒ is_ufl(t0,x,p)(ws′,ds′,ts)(ℓ)
```

Annotations

- The "do not exists" clauses shall ensure both largest sets of up front licenses over appropriate time spans, works, and devices and that there is no "junk".
- Otherwise we leave it to the reader to decipher the formulas.

```
value
  M(mkFR(t,x,p,ws,ds)) ≡
    {let ls = [ ti↦rndr(w,d)|ti:T,w:W,d:D•ti ∈ ts∧w ∈ ws′∧d ∈ ds′ ] in
     {(ti,ls(ti))|ti:T•ti ∈ dom ls} end
     ∪{(tn′,pay(x))}
    |n:Nat,tn′:T,ts:T-set,ws′:W-set,ds′:D-set•
    card ts=n∧ws′⊆ws∧ds′⊆ds∧t0≤min(ts)∧ max(ts)<tn′≤t0+p}
```

```
value
  M(mkPU(t,x,p,ws,ds)) ≡
    {let ls = [ ti↦rndr(w,d)|ti:T,w:W,d:D•ti ∈ ts∧w ∈ ws′∧d ∈ ds′ ] in
     {(ti,ls(ti))|ti:T•ti ∈ dom ls} end
     ∪ {(tn′,pay(n∗x))}
    |n:Nat,tn′:T,ts:T-set,ws′:W-set,ds′:D-set•
    card ts=n∧ws′⊆ws∧ds′⊆ds∧t0≤min(ts)∧ max(ts)<tn′≤t0+p}

value
  M(mkU(te,x,p,ws,ds)) ≡ ∪{M(mkUF(t0,x,p,ws,ds))|t0:T•t0≤te}

  M(mkNCMF((t,x,p,ws,ds),m)) ≡ UpFront^Δ((t,x,p,ws,ds),m)

  M(mkCMF((t,x,p,ws,ds),m)) ≡ UpFront^▷((t,x,p,ws,ds),m)

value
  Δ: L* → L
  Δ(ll) ≡
    case ll of
      ⟨l⟩ → l,
      ⟨l⟩^ll′ → {r ∪ r′|r,r′:R • r ∈ l ∧ r′ ∈ Δ(ll′)}
    end

  ▷: L* → L
  ▷(ll) ≡
    case ll of
      ⟨l⟩ → l,
      ⟨l⟩^ll′ → {r,r ∪ r′|r,r′:R • r ∈ l ∧ r≠{} ∧ r′ ∈ ▷(ll′)}
    end

value
  UpFront^Δ: Basics × Nat → L
  UpFront^Δ((t,x,p,ws,ds),m) ≡
    Δ(⟨M(mkUF(ti,x,p,ws,ds))|i:Nat,ti:T•i:{0..m−1}∧ti=t0+i×p⟩)

  UpFront^▷: Basics × Nat → L
  UpFront^▷((t,x,p,ws,ds),m) ≡
    ▷(⟨M(mkUF(ti,x,p,ws,ds))|i:Nat,ti:T•i:{0..m−1}∧ti=t0+i×p⟩)
```

A.7 A License Language

```
type
  DRExpr = Time × Repetition × Payment × WorksDevices
```


© Dines Bjørner 2006, Version 3, Fredsvej 11, DK-2840 Holte, Denmark

Time == mkAt(t:T) | mkUntil(t:T)
Repetition == mkFor(p:I) | mkRepeat(m:**Nat**,p:I) | mkUpTo(m:**Nat**,p:I)
Payment = P × Kind
Kind == upfront | flatrate | peruse
WorksDevices = W-**set** × D-**set**

value
M0(mkAt(t),r,(x,k),wds) ≡ M1(r,(x,k),wds)(t)
M0(mkUntil(t),r,(x,k),wds) ≡ ∪{M1(r,(x,k),wds)(t′)|t′:T•t′<t}

M1(mkFor(p),(x,k),wds)(t) ≡ M2(x,k)(t,p)
M1(mkRepeat(m,p),(x,k),wds)(t) ≡
  ▷(⟨M2((x,k),wds)(ti,p)|i:**Nat**,ti:T•i:{0..m−1}∧ti=t0+i×p⟩)
M1(mkUpTo(m,p),(x,k),wds)(t) ≡
  Δ(⟨M2((x,k),wds)(ti,p)|i:**Nat**,ti:T•i:{0..m−1}∧ti=t0+i×p⟩)

M2((x,upfront),(ws,ds))(t,p) ≡ M(mkUF(t,x,p,ws,ds))
M2((x,upfront),(ws,ds))(t,p) ≡ M(mkFR(t,x,p,ws,ds))
M2((x,upfront),(ws,ds))(t,p) ≡ M(mkPU(t,x,p,ws,ds))

A.8 End of "Gunter" Paper

- On one hand:
  - An introduction to the standard view of digital rights licenses.
- On the other hand:
  - An illustration of
    - a pseudo-formal erroneous model

      versus

    - a correct formal presentation.
- Now we are ready
  - to study digital rights license languages in general.