Fast Compliance Checking in an OWL2 Fragment

Piero A. Bonatti
Università degli Studi di Napoli Federico II
piero.bonatti@unina.it

Abstract

We illustrate a formalization of data usage policies in a fragment of OWL2. It can be used to encode (i) a company's data protection policy, (ii) data subjects' consent to data processing, and (iii) part of the GDPR (the forthcoming European Data Protection Regulation). Then a company's policy can be checked for compliance with data subjects' consent and with part of the GDPR by means of subsumption queries. We provide a complete and tractable structural subsumption algorithm for compliance checking and prove the intractability of a natural generalization of the policy language.

1 Introduction

This work stems from the EU H2020 project SPECIAL[1], where semantic technologies are used to help companies in complying with the new European General Data Protection Regulation (GDPR).[2] In this project, data usage policies are encoded using a fragment of OWL2-DL and the main policy-related reasoning tasks are reduced to subsumption and concept consistency checking. Such tasks include - among others:

- *permission checking*: given an operation request, decide whether it is permitted;
- *compliance checking*: does a policy P_1 fulfill all the restrictions requested by policy P_2? (Policy comparison);
- *policy validation*: e.g. is the policy contradictory? Does a policy update strengthen or relax the previous policy?

Compliance checking is the predominant task in this project: the data usage policies of the industrial partners must be compared both with a (partial) formalization of the GDPR itself, and with the consent to the usage of personal data granted by each of the *data subjects* whose data are collected and processed by the company (that is called *data controller* in the GDPR). The number of data subjects (and their policies) can be as large as the number of customers of a major communication service provider. Moreover, in the absence of explicit consent, some data cannot be stored, even temporarily; so some of the project's use cases consist in checking storage permissions against a stream of incoming data points, at the rate of hundreds of thousands per minute. Then one of the crucial project tasks is the development of scalable reasoning procedures for reasoning in the policy fragment of OWL 2.

In this paper we illustrate this fragment and introduce a structural subsumption algorithm that is a promising starting point for scalable compliance checking. Sec. 2 recalls the basics of Description Logics (DL) needed in this work. In Sec. 3 we illustrate the encoding of data usage policies in OWL2. In Sec. 4 we describe the structural subsumption algorithm for the policy language, and a policy consistency checking method. We prove correctness and completeness of these algorithms and their tractability. Moreover, we prove the intractability of a slight generalization of the policy fragment. The paper is concluded by a discussion of the results, a comparison with related work, and a description of ongoing and future work. Some proofs have been moved to Appendix A to enhance readability.

2 Preliminaries on Description Logics

Here we report the basics needed for our work and refer the reader to [Baader *et al.*, 2003] for further details. The DL languages of our interest are built from countably infinite sets of concept names ($\mathsf{N_C}$), role names ($\mathsf{N_R}$), concrete feature names ($\mathsf{N_F}$), and concrete predicates ($\mathsf{N_P}$). An *interpretation* $\mathcal{I}$ is a structure $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a nonempty set, and the *interpretation function* $\cdot^\mathcal{I}$ is such that (i) $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ for all $A \in \mathsf{N_C}$; (ii) $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for all $R \in \mathsf{N_R}$; (iii) $f^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathsf{D}$ for all $f \in \mathsf{N_F}$.[3] The semantics of an n-ary predicate $p \in \mathsf{N_P}$ is a set of tuples $p^\mathsf{D} \subseteq (\Delta^\mathsf{D})^n$. In this paper we use $\Delta^\mathsf{D} = \mathbb{N}$ and unary concrete predicates $\mathtt{in}_{\ell,u}$, where $\ell, u \in \mathbb{N}$, such that $\mathtt{in}^\mathsf{D}_{\ell,u} = [\ell, u]$. To enhance readability we will abbreviate $\mathtt{in}_{\ell,u}(f)$ to $[\ell, u](f)$. Informally, $[\ell, u](f)$ means that the value of feature f belongs to the interval $[\ell, u]$.

Figure 1 shows some DL operators and their semantics, that extends $\cdot^\mathcal{I}$ to compound DL expressions. The last four lines illustrate some DL axioms; GCI stands for "general

[1] https://www.specialprivacy.eu/

[2] http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf

[3] Δ^D denotes the domain of the predicates in $\mathsf{N_P}$. We are assuming – for brevity – that there is one concrete domain. However, our framework can be immediately extended to multiple domains.

Name	Syntax	Semantics
bottom	$\bot$	$\bot^\mathcal{I} = \emptyset$
intersection	$C \sqcap D$	$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
union	$C \sqcup D$	$(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$
restriction	$\exists R.C$	$\{d \in \Delta^\mathcal{I} \mid \exists (d,e) \in R^\mathcal{I} : e \in C^\mathcal{I}\}$
complement	$\neg C$	$(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$
concrete constraints	$p(f_1,..,f_n)$	$\{x{\in}\Delta^\mathcal{I} \mid \exists\vec{v}{\in}(\Delta^\mathsf{D})^n.\,(x,v_i) \in f_i^\mathcal{I}$ $(1 \leq i \leq n)$ and $\vec{v} \in p^\mathsf{D}\}$
GCI	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
disjointness	$\mathsf{disj}(C, D)$	$C^\mathcal{I} \cap D^\mathcal{I} = \emptyset$
func	$\mathsf{func}(R)$	$R^\mathcal{I}$ is a partial function
range	$\mathsf{range}(R, C)$	$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times C^\mathcal{I}$

Figure 1: Syntax and semantics of some DL constructs and axioms.

concept inclusion". An interpretation $\mathcal{I}$ *satisfies* an axiom α (equivalently, $\mathcal{I}$ is a *model* of α) if $\mathcal{I}$ satisfies the corresponding semantic condition in Fig. 1.

A *knowledge base* $\mathcal{K}$ is a finite set of DL axioms. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) if $\mathcal{I}$ satisfies all the axioms in $\mathcal{K}$. We say that "$\mathcal{K}$ *entails* an axiom α" – in symbols, $\mathcal{K} \models \alpha$ – if all models of $\mathcal{K}$ satisfy α.

A *pointed interpretation* is a pair $(\mathcal{I}, d)$ where $d \in \Delta^\mathcal{I}$. We say $(\mathcal{I}, d)$ *satisfies* a concept C ($(\mathcal{I}, d) \models C$) iff $d \in C^\mathcal{I}$.

3 Encoding Usage Policies in OWL2-DL

The GDPR states that the personal data of a data subject S can be collected, stored, processed and shared by an organization (call *data controller* in the GDPR) only after S consents to such usage. This norm admits only a few exceptions, e.g. when personal data handling is required by law or is for the public good. Anonymized data are not subject to the GDPR and can be freely stored and processed.

In project SPECIAL, compliance with the GDPR is supported by (i) formalizing consent and the data controllers' data usage policies (called *business policies*), and (ii) formalizing selected parts of the GDPR, mainly related to consent and data transfer.

The aspects of data usage that must be specified in consent requests – and preserved after approval – are clearly indicated in the GDPR and in the available guidelines[4]:

- reasons for data processing (purpose);
- which data categories are involved;
- what kind of processing;
- which third parties data are distributed to (recipients);
- countries in which the data will be stored (location);
- time limits for removal of data (duration).

The above properties characterize a *usage policy*. In SPECIAL we adopt a direct encoding of usage policies in description logics. The simplest possible policies have the form:

$$\begin{aligned}&\exists \mathit{purp}.P \sqcap \exists \mathit{data}.D \sqcap \exists \mathit{proc}.O \sqcap \exists \mathit{recip}.R \sqcap \\ &\exists \mathit{storage}(\exists \mathit{loc}.L \sqcap \exists \mathit{dur}.T)\,.\end{aligned} \tag{1}$$

The classes P, D, O, etc. are defined in suitable vocabularies (ontologies) that specify also their mutual relationships (inclusion, disjointness etc.). SPECIAL temporarily adopts simple vocabularies derived from ODRL[5] (for describing processing) and P3P[6] (for the other properties), while longer-term standardization activities are in progress. Currently, such taxonomies are encoded with simple inclusions $A \sqsubseteq B$ and disjointness constraints $\mathsf{disj}(A, B)$, where A, B are concept *names*.[7] Duration is represented internally as an interval of integers $[t_1, t_2]$. All of the roles in (1) are functional.

The above formalization represents the class of all data usage modalities whose properties are instances of the respective vocabulary terms, that is, when the data subject consents to (1), then she authorizes all of its instances. For example if $D = \mathrm{DemographicData}$ then the data subject authorizes the use of her name, address, age, income, etc. as specified by the other policy properties.

It frequently happens that the data controller intends to use different data categories in different ways, according to their usefulness and sensitivity, so consent requests comprise multiple simple usage policies like (1) (one for each usage type). The intended meaning is that consent is requested for all the instances of all those policies; accordingly, such a compound policy is formalized as the union of its components and a *full* (usage) policy is a concept

$$P_1 \sqcup \ldots \sqcup P_n \tag{2}$$

where each P_i has the form (1).

Example 1 A company – call it BeFit – sells a wearable fitness appliance and wants (i) to process biometric data (stored in the EU) for sending health-related advice to a customer, and (ii) share her location data with her friends. Location data are kept for a minimum of one year but no longer than 5; biometric data are kept for an unspecified amount of time. In order to do all this legally, BeFit needs consent from its customers. The internal (formalized) consent description would look as follows:

$$\begin{aligned}&(\exists \mathit{purp}.\mathrm{FitnessRecommendation} \sqcap \\ &\ \exists \mathit{data}.\mathrm{BiometricData} \sqcap \\ &\ \exists \mathit{proc}.\mathrm{Analytics} \sqcap \\ &\ \exists \mathit{recip}.\mathrm{BeFit} \sqcap \\ &\ \exists \mathit{storage}.\mathit{loc}.\mathrm{EU}) \\ &\sqcup \\ &(\exists \mathit{purp}.\mathrm{SocialNetworking} \sqcap \\ &\ \exists \mathit{data}.\mathrm{LocationData} \sqcap \\ &\ \exists \mathit{proc}.\mathrm{Transfer} \sqcap \\ &\ \exists \mathit{recip}.\mathrm{DataSubjFriends} \sqcap \\ &\ \exists \mathit{storage}.(\mathit{loc}.\mathrm{EU} \sqcap [y_1, y_5](\mathit{dur}))\,.\end{aligned} \tag{3}$$

Here y_1 and y_5 are the internal, integer representation of one year and five years. If "HeartRate" is a subclass of "BiometricData" and "ComputeAvg" a subclass of "Analytics", then the above consent allows BeFit to compute the average heart rate of the data subject for sending her fitness

[4] E.g. see the 'Records should contain' section in http://ec.europa.eu/justice/smedataprotect/index_en.htm

[5] https://www.w3.org/TR/odrl/

[6] http://www.w3.org/TR/P3P11

[7] In the journal version we will show how to support general $\mathcal{EL}$ knowledge bases.

recommendations. The purpose could be further detailed by saying how the data subject is contacted, e.g. by replacing the first line with $\exists purp.(\text{FitnessRecommendation} \sqcap \exists contact.\text{SMS})$. $\blacksquare$

The usage policies that are actually applied by the data controller in its business processes are called *business policies* and include a description of data usage of the form (1). Additionally, each business policy describes the obligations that must be fulfilled, related to the specified usage. For example, if the data category includes personal data, then, according to the GDPR, the business policy should have the additional conjuncts

$$\begin{aligned}&\exists duty.\text{GetConsent} \sqcap \exists duty.\text{GiveAccess} \sqcap\\ &\exists duty.\text{RectifyOnRequest} \sqcap\\ &\exists duty.\text{DeleteOnRequest}\end{aligned} \tag{4}$$

that model the obligations related to the data subjects's rights prescribed by the GDPR.

Such business policies are an abstract description of a business process, highlighting the aspects related to compliance with the GDPR and with data subjects' consent. Similarly to consent, a business policy may be a union $BP_1 \sqcup \ldots \sqcup BP_n$ of simple business policies BP_i of the form $(1) \sqcap (4)$.

In some contexts, business policies can be assembled in a semi-automated way by tagging:

- data sources and their schemas with vocabulary terms specifying the data source's location and the data categories it stores, as well as the third parties that may access those data;
- programs or processes with vocabulary terms describing the nature of processing and its purpose, as well as the third parties to which the results are transferred.

In order to check whether a business process complies with the consent given by a data subject S it suffices to check whether the corresponding business policy BP is subsumed by the consent policy of S, denoted by CP_S (in symbols, $BP \sqsubseteq CP_S$). This subsumption is checked against a knowledge base that encodes type restrictions related to policy properties and the corresponding vocabularies, i.e. subclass relationships, disjointness constraints, functionality restrictions, domain and range restrictions, and the like.

In order to verify that all the required obligations are fulfilled by a business process (as abstracted by the business policy), selected parts of the GDPR are formalized with concepts like:

$$\begin{aligned}&(\exists duty.\text{GetConsent} \sqcap \exists duty.\text{GiveAccess} \sqcap \ldots) \sqcup\\ &\exists data.\text{Anonymous} \sqcup\\ &\exists purp.\text{LawRequirement} \sqcup \ldots\end{aligned} \tag{5}$$

(that states that a business policy should either satisfy (4) or concern anonymous data, or some of the exceptional cases such as particular law requirements), and

$$\begin{aligned}&\exists storage.loc.\text{EU} \sqcup\\ &\exists storage.loc.\text{EULike} \sqcup \ldots\end{aligned} \tag{6}$$

(stating that data should remain within the EU, or countries that adopt similar data protection regulations).

Then a business policy BP can be checked for compliance with the formalized parts of the GDPR by checking whether the aforementioned knowledge base entails that BP is subsumed by all the above concepts.

Example 2 The following business policy complies with the consent-related obligations formalized in (5) since it is subsumed by it:

$$\begin{aligned}&(\exists purp.\text{FitnessRecommendation} \sqcap\\ &\ \exists data.\text{BiometricData} \sqcap\\ &\ \exists proc.\text{Analytics} \sqcap\\ &\ \exists recip.\text{BeFit} \sqcap\\ &\ \exists storage.loc.\text{EU} \sqcap\\ &\ \exists duty.\ldots(4)\ldots)\\ &\sqcup\\ &(\exists purp.\text{Sell} \sqcap\\ &\ \exists data.\text{Anonymous} \sqcap\\ &\ \exists proc.\text{Transfer} \sqcap\\ &\ \exists recip.\text{ThirdParty})\,.\end{aligned} \tag{7}$$

In particular, the two disjuncts of (7) are subsumed by the first two lines of (5), respectively. $\blacksquare$

Based on the above discussion, we are now ready to specify a fragment of OWL 2 called $\mathcal{PL}$ (*policy logic*) that covers – and slightly generalizes – the language outlined above.

Definition 1 (Policy logic $\mathcal{PL}$) *A* $\mathcal{PL}$ knowledge base $\mathcal{K}$ *is a set of axioms of the following kinds:*

- $\mathsf{func}(R)$ *where* R *is a role name or a concrete feature;*
- $\mathsf{range}(S, A)$ *where* S *is a role and* A *a concept name;*
- $A \sqsubseteq B$ *where* A, B *are concept names;*
- $\mathsf{disj}(A, B)$ *where* A, B *are concept names.*

A simple $\mathcal{PL}$ concept *has the form:*

$$A_1 \sqcap \ldots \sqcap A_n \sqcap E_1 \sqcap \ldots \sqcap E_m \sqcap C_1 \sqcap \ldots \sqcap C_t \tag{8}$$

where each A_i *is either a concept name or* $\bot$, *each* E_j *is an existential restriction* $\exists R.C$ *such that* R *is a role and* C *a simple* $\mathcal{PL}$ *concept, and each* C_k *is a concrete constraint* $[l, u](f)$. *A* (full) $\mathcal{PL}$ concept *is a union* $D_1 \sqcup \ldots \sqcup D_n$ *of simple* $\mathcal{PL}$ *concepts.* $\mathcal{PL}$*'s* subsumption queries *are expressions* $C \sqsubseteq D$ *where* C, D *are (full)* $\mathcal{PL}$ *concepts.*

4 Compliance Checking and its Complexity

In this section we introduce a structural subsumption algorithm for deciding whether $\mathcal{K} \models C \sqsubseteq D$, for all given $\mathcal{PL}$ KB $\mathcal{K}$ and all $\mathcal{PL}$ subsumptions $C \sqsubseteq D$. We start by introducing an auxiliary algorithm for *elementary* subsumptions, that are $\mathcal{PL}$ subsumptions $C \sqsubseteq D$ where both C and D are simple, and $C \sqsubseteq D$ is *interval safe*, that is, for all constraints $[\ell, u](f)$ occurring in C and $[\ell', u'](f')$ occurring in D, either $[\ell, u] \subseteq [\ell', u']$, or $[\ell, u] \cap [\ell', u'] = \emptyset$.[8]

The structural subsumption algorithm (Algorithm 1, hereafter STS) takes as input a $\mathcal{PL}$ KB $\mathcal{K}$ and an elementary $\mathcal{PL}$

[8] We will show later that all subsumptions can be turned into equivalent interval safe subsumptions.

subsumption $C \sqsubseteq D$, where C is normalized w.r.t. $\mathcal{K}$ using the rewrite rules in Table 1. There, $\sqsubseteq^*$ denotes the reflexive and transitive closure of $\{(A, B) \mid (A \sqsubseteq B) \in \mathcal{K}\}$. By a straightforward case analysis we obtain Proposition 1:

```
Algorithm 1: STS($\mathcal{K}, C \sqsubseteq D$)
  Input: $\mathcal{K}$ and an elementary $C \sqsubseteq D$ where $C$ is normalized
  Output: true if $\mathcal{K} \models C \sqsubseteq D$,   false otherwise
  Note: Below, by $C = C' \sqcap C''$ we mean that either $C = C'$ or
  $C'$ is a conjunct of $C$ (possibly not the first one)
1 begin
2     if $C = \bot$ then return true
3     if $D = A$, $C = A' \sqcap C'$ and $A' \sqsubseteq^* A$ then return true
4     if $D = [l, u](f)$ and $C = [l', u'](f) \sqcap C'$ and $l \leq l'$ and
      $u' \leq u$ then return true
5     if $D = \exists R.D'$, $C = (\exists R.C') \sqcap C''$ and
      STS($\mathcal{K}$, $C' \sqsubseteq D'$) then return true
6     if $D = D' \sqcap D''$, STS($\mathcal{K}, C \sqsubseteq D'$), and
      STS($\mathcal{K}, C \sqsubseteq D''$) then return true
7     else return false
8 end
```

Proposition 1 *The rewrite rules in Table 1 preserve concept equivalence w.r.t. $\mathcal{K}$, i.e. if $C \rightsquigarrow C'$ then $\mathcal{K} \models C \equiv C'$.*

In order to prove that STS is complete w.r.t. elementary subsumptions we need a canonical counterexample to invalid subsumptions.

Definition 2 *Let $C \neq \bot$ be a simple $\mathcal{PL}$ concept normalized w.r.t. $\mathcal{K}$. A* canonical model *of C is a pointed interpretation $(\mathcal{I}, d)$ defined as follows, by recursion on the nesting level of existential restrictions. Hereafter we call a subconcept of C "top level" if it does not occur within the scope of $\exists$.*

a. If $C = \left(\prod_{i=1}^{n} A_i\right) \sqcap \left(\prod_{j=1}^{t} C_j\right)$ (i.e. C has no existential restrictions), then let $\mathcal{I} = \langle\{d\}, \cdot^{\mathcal{I}}\rangle$ where

- *$A^{\mathcal{I}} = \{d\}$ iff for some $i = 1, \ldots, n$, $A_i \sqsubseteq^* A$;*
- *if $C_j = [l, u](f)$, add (d, u) to $f^{\mathcal{I}}$ $(1 \leq j \leq t)$;*
- *all the other predicates are empty.*

b. If the top-level existential restrictions of C are $\exists R_i.D_i$ $(i = 1, \ldots, m)$, then let $(\mathcal{I}_i, d_i)$ be a canonical model of D_i, for each $i = 1, \ldots, m$. Assume w.l.o.g. that all such models are mutually disjoint and do not contain d (if necessary, their elements can be replaced). Define an auxiliary interpretation $\mathcal{J}$ as follows:

- *$\Delta^{\mathcal{J}} = \{d, d_1, \ldots, d_m\}$;*
- *for all concept names A such that for some top-level A_i in C, $A_i \sqsubseteq^* A$, let $A^{\mathcal{J}} = \{d\}$; all other concept names are empty;*
- *for each top-level constraint $[l, u](f)$ in C, add (d, u) to $f^{\mathcal{J}}$;*
- *for each top-level restriction $\exists R_i.D_i$ in C, add a pair (d, d_i) to $R_i^{\mathcal{I}}$;*
- *there are no other pairs in roles and features.*

Finally let $\mathcal{I}$ be the union of $\mathcal{J}$ and all $\mathcal{I}_i$, that is

$$\begin{array}{rcll} \Delta^{\mathcal{I}} & = & \Delta^{\mathcal{J}} \cup \bigcup_i \Delta^{\mathcal{I}_i} & \\ A^{\mathcal{I}} & = & A^{\mathcal{J}} \cup \bigcup_i A^{\mathcal{I}_i} & (A \in \mathsf{N_C}) \\ R^{\mathcal{I}} & = & R^{\mathcal{J}} \cup \bigcup_i R^{\mathcal{I}_i} & (R \in \mathsf{N_R} \cup \mathsf{N_F})\,. \end{array}$$

The canonical model is $(\mathcal{I}, d)$.

Note that each C has a unique canonical model up to isomorphism. The canonical model actually satisfies $\mathcal{K}$ and C:

Lemma 1 *If C is a simple $\mathcal{PL}$ concept normalized w.r.t. $\mathcal{K}$, and $C \neq \bot$, then the canonical model $(\mathcal{I}, d)$ of C enjoys the following properties:*

a. $\mathcal{I} \models \mathcal{K}$;

b. $(\mathcal{I}, d) \models C$.

Moreover, the canonical model of C characterizes *all* the valid elementary subsumptions whose left-hand side is C:

Lemma 2 *If $C \sqsubseteq D$ is elementary, and C is normalized w.r.t. $\mathcal{K}$, then $\mathcal{K} \models C \sqsubseteq D$ iff $(\mathcal{I}, d) \models D$, where $(\mathcal{I}, d)$ is the canonical model of C.*

Basically, STS decides whether $(\mathcal{I}, d) \models D$.

Lemma 3 *If $C \sqsubseteq D$ is elementary, $C \neq \bot$, and C is normalized w.r.t. $\mathcal{K}$, then* STS$(\mathcal{K}, C \sqsubseteq D) =$ `true` *iff $(\mathcal{I}, d) \models D$, where $(\mathcal{I}, d)$ is the canonical model of C.*

The correctness and completeness of STS easily follow:

Theorem 2 *If $C \sqsubseteq D$ is elementary and C is normalized w.r.t. $\mathcal{K}$, then* STS$(\mathcal{K}, C \sqsubseteq D) =$ `true` *iff $\mathcal{K} \models C \sqsubseteq D$.*

Proof. There are two possibilities. If $C = \bot$, then clearly $\mathcal{K} \models C \sqsubseteq D$ and STS$(\mathcal{K}, C \sqsubseteq D) =$ `true` (line 2 of STS), so the theorem holds. If $C \neq \bot$, then theorem follows immediately from lemmas 2 and 3. ∎

Now, through Lemma 2, subsumption checks over *full* $\mathcal{PL}$ concepts can be reduced to elementary subsumptions.

Theorem 3 *For all interval-safe $\mathcal{PL}$ subsumption queries $\sigma = \left(C_1 \sqcup \ldots \sqcup C_m \sqsubseteq D_1 \sqcup \ldots \sqcup D_n\right)$ such that each C_i is normalized w.r.t. $\mathcal{K}$, the entailment $\mathcal{K} \models \sigma$ holds iff for all $i \in [1, m]$ there exists $j \in [1, n]$ such that $\mathcal{K} \models C_i \sqsubseteq D_j$.*

Proof. By simple logical inferences, these two facts hold: (i) $\mathcal{K} \models \sigma$ iff $\mathcal{K} \models C_i \sqsubseteq \bigsqcup_{j=1}^{n} D_j$ holds for all $i \in [1, m]$, (ii) if $\mathcal{K} \models C_i \sqsubseteq D_j$ holds for some $j \in [1, n]$, then $\mathcal{K} \models C_i \sqsubseteq \bigsqcup_{j=1}^{n} D_j$. So we are only left to show the converse of (ii): assuming that $\mathcal{K} \not\models C_i \sqsubseteq D_j$ for all $j \in [1, n]$, we shall prove that $\mathcal{K} \not\models C_i \sqsubseteq \bigsqcup_{j=1}^{n} D_j$.

By assumption and Lemma 2, the canonical model $(\mathcal{I}, d)$ of C_i is such that $(\mathcal{I}, d) \models \neg D_j$ for all $j \in [1, n]$. Therefore $(\mathcal{I}, d) \models \neg \bigsqcup_{j=1}^{n} D_j$. Then $\mathcal{K} \not\models C_i \sqsubseteq \bigsqcup_{j=1}^{n} D_j$ follows by noting that $(\mathcal{I}, d)$ satisfies both $\mathcal{K}$ and C_i by Lemma 1. ∎

Theorem 3 directly yields an obvious polynomial-time algorithm for checking $\mathcal{PL}$ subsumptions: it suffices to check that for each C_i there exists D_j such that $\mathcal{K} \models C_i \sqsubseteq D_j$. So:

Theorem 4 *Interval-safe $\mathcal{PL}$ subsumption queries can be answered in polynomial time.*

1)	$\bot \sqcap D \rightsquigarrow \bot$	
2)	$\exists R.\bot \rightsquigarrow \bot$	
3)	$[l,u](f) \rightsquigarrow \bot$	if $l > u$
4)	$(\exists R.D) \sqcap (\exists R.D') \sqcap D'' \rightsquigarrow \exists R.(D \sqcap D') \sqcap D''$	if $\mathsf{func}(R) \in \mathcal{K}$
5)	$[l_1,u_1](f) \sqcap [l_2,u_2](f) \sqcap D \rightsquigarrow [\max(l_1,l_2), \min(u_1,u_2)](f) \sqcap D$	if $\mathsf{func}(f) \in \mathcal{K}$
6)	$\exists R.D \sqcap D' \rightsquigarrow \exists R.(D \sqcap A) \sqcap D'$	if $\mathsf{range}(R,A) \in \mathcal{K}$ and A not a conjunct of D
7)	$A_1 \sqcap A_2 \sqcap D \rightsquigarrow \bot$	if $A_1 \sqsubseteq^* A'_1$, $A_2 \sqsubseteq^* A'_2$, and $\mathsf{disj}(A'_1, A'_2) \in \mathcal{K}$

Table 1: Normalization rules w.r.t. $\mathcal{K}$. Intersections are treated as sets (the ordering of conjuncts and their repetitions are irrelevant).

Proof. Let σ be a $\mathcal{PL}$ query of the form illustrated in Theorem 3. By that theorem, after normalizing the C_i, it suffices to call $\mathsf{STS}(\mathcal{K}, C_i \sqsubseteq D_j)$ at most $m \times n$ times. Each such calls scans C_i for each subconcept of D_j, searching for a matching concept. Matching may require to visit the hierarchy $\sqsubseteq^*$, so the cost of each call is $O(|D_j| \cdot |C_i| \cdot |\mathcal{K}|) = O(|\sigma|^2 \cdot |\mathcal{K}|)$. Then the overall cost of a naive implementation – excluding normalization – is $O(|\sigma|^4 \cdot |\mathcal{K}|)$. Normalization is $O(|\sigma|^2 \cdot |\mathcal{K}|)$, due to the cost of checking pairwise disjointness of concept names, so it is dominated by the cost of STS's runs. ▮

We are left to discuss the interval-safety prerequisite needed by Theorem 3.[9] It can be satisfied as follows, with a *preliminary interval normalization phase* of the query $C \sqsubseteq D$.

For each $[\ell, u](f)$ in C, let $x_1 < x_2 < \cdots < x_r$ be the integers that occur as interval endpoints in D and belong to $[\ell, u]$. Let $x_0 = \ell$ and $x_{r+1} = u$ and replace $[\ell, u](f)$ with the equivalent concept

$$\bigsqcup_{i=0}^{r} \big([x_i, x_i](f) \sqcup [x_i + 1, x_{i+1} - 1](f)\big) \sqcup [x_{r+1}, x_{r+1}](f). \quad (9)$$

Then use distributivity of $\sqcap$ over $\sqcup$ and the equivalence $\exists R.(C_1 \sqcup C_2) \equiv \exists R.C_1 \sqcup \exists R.C_2$ to move all occurrences of $\sqcup$ to the top level. Denote the result of this interval normalization phase with C^*.

Example 3 Let $C = [1, 9](f) \sqcap A$ and $D = [5, 12](f)$. Then $r = 1$ and $x_0 = 1$, $x_1 = 5$, $x_3 = 9$ (12 falls outside $[1, 9]$ and is ignored). Concept $[1, 9](f)$ is split as follows: $[1, 1](f) \sqcup [2, 4](f) \sqcup [5, 5](f) \sqcup [6, 8](f) \sqcup [9, 9](f)$. Then $C^* = ([1, 1](f) \sqcap A) \sqcup ([2, 4](f) \sqcap A) \sqcup ([5, 5](f) \sqcap A) \sqcup ([6, 8](f) \sqcap A) \sqcup ([9, 9](f) \sqcap A)$. ▮

Readers may easily verify that

Proposition 5 *For all $\mathcal{PL}$ subsumption queries $C \sqsubseteq D$, C^* is equivalent to C and $C^* \sqsubseteq D$ is an interval-safe $\mathcal{PL}$ subsumption query.*

Clearly, interval normalization may inflate C exponentially, due to the application of distributivity (e.g. this happens with the concepts C and D in the proof of Theorem 7). We have to rely on the structure of policies to get a polynomial time bound: simple ($\sqcup$-free) policies have at most one, functional concrete feature (`dur`) so no combinatorial explosion occurs during pre-normalization. Accordingly—and more generally—the following proposition holds:

Proposition 6 *$\mathcal{PL}$ subsumption queries $C_1 \sqcup \ldots \sqcup C_m \sqsubseteq D$ can be answered in polynomial time if there exists a constant c such that for each C_i, the number of concrete features occurring in C_i is bounded by c.*

Note that C may contain any number of interval constraints, as m grows, because the bound applies to each C_i individually. This may seem unsatisfactory at a first glance. Unfortunately, no general polynomial-time algorithms exist (unless P=NP) because the interplay of interval constraints and $\sqcup$ makes unrestricted $\mathcal{PL}$ subsumption queries intractable:

Theorem 7 *Subsumption checking in $\mathcal{PL}$ is co*NP*-complete. The result holds even if the knowledge base is empty.*

We conclude this section by pointing out that the normalization rules in Table 1 can be used as a *policy validation* method, to check that a full $\mathcal{PL}$ concept is satisfiable.

Proposition 8 *Let $\mathcal{K}$ be a $\mathcal{PL}$ knowledge base.*

1. *A $\mathcal{PL}$ concept $C = C_1 \sqcup \ldots \sqcup C_n$ is unsatisfiable w.r.t. $\mathcal{K}$ iff $C_i \rightsquigarrow \bot$ for all $i \in [1, n]$.*
2. *$\mathcal{PL}$ concept satisfiability w.r.t. $\mathcal{K}$ can be checked in polynomial time.*

Proof. Point 1 follows immediately from Prop. 1 and Lemma 1. It is easy to see that normalization takes polynomial time, so Point 2 holds. ▮

5 Discussion and Future Work

The encoding of usage policies in OWL 2 – using the tractable fragment of the description logic $\mathcal{PL}$ introduced in this paper – is simple enough to reduce compliance checking to tractable structural subsumption tests. Such tests can be computed in polynomial time w.r.t. the size of the ontology and the size of the policies involved (i.e. company policies and formalized consent). For a fixed company policy and fixed vocabularies, the complexity of verifying whether the company policy complies with the explicit consent released by data subjects grows linearly with the size of formalized consent, so we expect compliance checking to scale well to large numbers of data subjects and articulated consent expressions. Tractability relies on the bound on the number of concrete features in policies, as unrestricted $\mathcal{PL}$ subsumption is coNP-complete.

We are planning – as part of the activities of the SPECIAL project – to implement and optimize the structural subsumption algorithm and perform scalability tests. The compliance checking task is well-suited for parallelization (e.g. the compliance tests with respect to different consent policies are all mutually independent and can be carried out in parallel). For this purpose we will leverage the Big Data Europe infrastructure adopted by SPECIAL. We are also going to integrate the structural subsumption algorithm with ELK, a specialized

[9] Theorem 2 can also be proved without assuming interval safety. The proof will be given in a forthcoming journal version.

reasoner for the tractable profile OWL2-EL [Kazakov *et al.*, 2013]. In this way, the vocabularies of data categories, purposes, locations, etc. can be expressed with OWL2-EL, instead of the limited axioms over concept names allowed in $\mathcal{PL}$ knowledge bases.

The real-time checking of storage consent for data streams, mentioned in the introduction, most likely requires further optimization. We foresee knowledge compilation techniques as a promising approach. Their feasibility can only be verified experimentally, and will be the subject of further work.

The previous encodings of policies in KR languages, such as [Uszok *et al.*, 2003; Kagal *et al.*, 2003; Bonatti *et al.*, 2010], focus on access control and trust management, rather than data usage control. Consequently, those languages lack the terms for expressing privacy-related and usage-related concepts. A more critical drawback is that the main reasoning task in those papers is permission checking; policy comparison (which is central to our work) is not considered. Both Rei and Protune [Kagal *et al.*, 2003; Bonatti *et al.*, 2010] support logic program rules. If rules are recursive then policy comparison is generally undecidable, otherwise it is NP-hard. The comparison of $\mathcal{PL}$ policies, instead, is tractable (Proposition 6). Similarly, KAoS [Uszok *et al.*, 2003] is based on a DL that, in general, is not tractable, and supports role-value maps that in general make reasoning undecidable (see [Baader *et al.*, 2003], Chap. 5). The authors do not discuss how to avoid this issue. A DL-based approach that does not suffer from potential undecidabiblity and stresses the importance of policy comparison and other nonstandard reasoning tasks is [Kolovski *et al.*, 2007]. However, tractable fragments and scalability lie beyond the scope of that paper.

Our tractability and intractability results do not follow directly from any previous work. The most expressive language enjoying a complete structural subsumption algorithm – to the best of our knowledge – is the description logic underlying CLASSIC [Borgida and Patel-Schneider, 1994], that supports neither concept unions ($\sqcup$) nor qualified existential restrictions ($\exists R.C$). If unions were added, then subsumption checking would immediately become coNP-hard (unless concrete domains were restricted) for the same reasons why unrestricted subsumption checking is coNP-hard in $\mathcal{PL}$ (cf. Theorem 7). On the other hand, CLASSIC additionally supports qualified universal restrictions (that strictly generalize $\mathcal{PL}$'s range restrictions), number restrictions, and role-value maps, therefore it is not comparable to $\mathcal{PL}$.

$\mathcal{PL}$ partially intersects the $\mathcal{EL}$ family of tractable DL, too. In particular, $\mathcal{EL}^{++}$ supports $\bot$ (hence it can express disj), $\sqcap$, qualified existential restrictions and concrete domains. It is known that role range restrictions can be added without affecting tractability [Baader *et al.*, 2008], and that union can be supported in queries. However, it has been proved that functional roles make $\mathcal{EL}$ EXP-complete [Baader *et al.*, 2005]. A tractability result for empty TBoxes can be found in [Haase and Lutz, 2008]; however, in the same paper, it is proved that even with acyclic TBoxes, subsumption is coNP-complete. Therefore, $\mathcal{PL}$ is incomparable with the known tractable logics in the $\mathcal{EL}$ family. Moreover, our coNP-completeness result is not entailed by the intractability result for $\mathcal{EL}$ with functional roles.

A Proofs

Proof of Lemma 1 By induction on the maximum nesting level ℓ of C's existential restrictions.

If $\ell = 0$ (i.e. there are no existential restrictions) then $(\mathcal{I}, d) \models C$ by construction (cf. Def. 2.a). Disjointness axioms are satisfied, otherwise normalization would make $C = \bot$ (contradicting the hypotheses). Inclusion axioms are satisfied due to the first and third bullets of Def. 2.a. Moreover, $\mathcal{I}$ trivially satisfies all the functionality and range assertions in $\mathcal{K}$ since all roles are empty. It follows that the lemma holds for the base case.

Now suppose that $\ell > 0$. By induction hypothesis (I.H), we have that all the submodels $(\mathcal{I}_i, d_i)$ used in Def. 2.b satisfy both D_i and $\mathcal{K}$. Then it is immediate to see that $(\mathcal{I}, d) \models C$ by construction. We are only left to prove that $\mathcal{I}$ satisfies all axioms α in $\mathcal{K}$.

If $\alpha = \mathsf{func}(R)$, then rewrite rules 4) and 5) make sure that C contains at most one existential restriction for R, so $\mathcal{J}$ satisfies α. Since all $\mathcal{I}_i$ satisfy α by I.H. (induction hypothesis), $\mathcal{I}$ satisfies α, too.

If $\alpha = \mathsf{range}(R, A)$, then rule 6) makes sure that for each top-level $\exists R_i.D_i$ in C, $D_i \equiv D'_i \sqcap A$. Then, by I.H., $(I_i, d_i) \models A$.

If $\alpha = \mathsf{disj}(A, B)$, and α were not true in $\mathcal{J}$, then rule 7) would make $C = \bot$ (a contradiction); the other parts of $\mathcal{I}$, i.e. $(\mathcal{I}_i, d_i)$, satisfy α by I.H.

Finally, inclusions are satisfied by construction, cf. the second bullet of Def. 2.b. It follows that $\mathcal{I}$ satisfies α. ∎

Proof of Lemma 2 (Only If part) Assume that $\mathcal{K} \models C \sqsubseteq D$. By Lemma 1.a, $\mathcal{I} \models \mathcal{K}$, so $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Moreover, by Lemma 1.b, $d \in C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Therefore $(\mathcal{I}, d) \models D$.

(If part) Assume that $(\mathcal{I}, d) \models D$. We are going to prove that $\mathcal{K} \models C \sqsubseteq D$ by structural induction on D.

If $D = A$ (a concept name), then $d \in A^{\mathcal{I}}$. Then, by construction of $\mathcal{I}$, $C = A_i \sqcap C'$ (up to top-level subconcept reordering), and $A_i \sqsubseteq^* A$. These two facts, respectively, imply $\models C \sqsubseteq A_i$ and $\mathcal{K} \models A_i \sqsubseteq A$, hence $\mathcal{K} \models C \sqsubseteq D$.

If $D = D_1 \sqcap D_2$, then by I.H. $\mathcal{K} \models D_i$ $(i = 1, 2)$, hence $\mathcal{K} \models C \sqsubseteq D$.

If $D = \exists R.D_1$, then for some $d_i \in \Delta^{\mathcal{I}}$, $(d, d_i) \in R^{\mathcal{I}}$ and $(\mathcal{I}_i, d_i) \models D_1$, where $(\mathcal{I}_i, d_i)$ (by construction of $\mathcal{I}$) is the canonical model of a top-level restriction $\exists R.C_1$ in C. It follows (using the I.H.) that $\models C \sqsubseteq \exists R.C_1$ and $\mathcal{K} \models C_1 \sqsubseteq D_1$, hence $\mathcal{K} \models C \sqsubseteq D$.

If $D = [\ell, u](f)$, then for some $u' \in [\ell, u]$, $(d, u') \in f^{\mathcal{I}}$. By construction of $\mathcal{I}$, C must contain a top-level constraint $[\ell', u'](f)$, and by interval safety (as $C \sqsubseteq D$ is elementary), $[\ell', u'] \subseteq [\ell, u]$. Then $\models C \sqsubseteq D$. ∎

Proof of Lemma 3 By structural induction on D. If $D = A$ (a concept name), then $\mathsf{STS}(\mathcal{K}, C \sqsubseteq D) = \mathtt{true}$ iff C has a top-level subconcept A_i such that $A_i \sqsubseteq^* A$. By def. of $\mathcal{I}$, this holds iff $d \in A^{\mathcal{I}}$, that is, $(\mathcal{I}, d) \models D$. This proves the base case.

If $D = D_1 \sqcap D_2$, then the lemma follows easily from the induction hypothesis.

If $D = \exists R.D_1$, then $\mathsf{STS}(\mathcal{K}, C \sqsubseteq D) = \mathtt{true}$ iff (i) C has a top-level subconcept $\exists R.C_1$, and (ii) $\mathsf{STS}(\mathcal{K}, C_1 \sqsubseteq D_1) =$

true. Moreover, by def. of $\mathcal{I}$,$(\mathcal{I}, d) \models D$ iff (i) holds and (ii') $(\mathcal{I}_i, d_i) \models D_1$, where $(\mathcal{I}_i, d_i)$ is the canonical model of C_1. By I.H., (ii) is equivalent to (ii'), so the lemma holds.

If $D = [\ell, u](f)$, then $\mathsf{STS}(\mathcal{K}, C \sqsubseteq D) = \mathtt{true}$ iff C has a top-level subconcept $[\ell', u'](f)$ such that $[\ell', u'] \subseteq [\ell, u]$. This implies (by def. of $\mathcal{I}$) that $(d, u') \in f^{\mathcal{I}}$ and $u' \in [\ell, u]$, that is, $(\mathcal{I}, d) \models D$. Conversely, if $(d, u') \in f^{\mathcal{I}}$ and $u' \in [\ell, u]$, then C must have a top-level subconcept $[\ell', u'](f)$ such that also $\ell' \in [\ell, u]$ (by interval safety, that holds by the hypothesis that $C \sqsubseteq D$ is elementary). Then $\mathsf{STS}(\mathcal{K}, C \sqsubseteq D) = \mathtt{true}$. ∎

Proof of Theorem 7 Hardness is proved by reducing 3SAT to the complement of subsumption. Let S be a given set of clauses $c_i = L_{i1} \vee L_{i2} \vee L_{i3}$ $(1 \leq i \leq n)$ where each L_{ij} is a literal. We are going to use the propositional symbols $p_1, \ldots, p_m$ occurring in S as feature names in $\mathcal{PL}$ concepts, and define a subsumption $C \sqsubseteq D$ that is valid iff S is unsatisfiable. Let $C = \big([0,1](p_1) \sqcap \ldots \sqcap [0,1](p_m)\big)$ and $D = \bigsqcup_{i=1}^{n} \big(\tilde{L}_{i1} \sqcap \tilde{L}_{i2} \sqcap \tilde{L}_{i3}\big)$, where each $\tilde{L}_{ij}$ encodes the complement of L_{ij} as follows:

$$\tilde{L}_{ij} = \begin{cases} [0,0](p_k) & \text{if } L_{ij} = p_k\,, \\ [1,1](p_k) & \text{if } L_{ij} = \neg p_k\,. \end{cases}$$

The correspondence between the propositional interpretations I of S and the interpretations $\mathcal{J}$ of $C \sqsubseteq D$ is the following.

Given I and an arbitrary element d, define $\mathcal{J} = \langle \{d\}, \cdot^{\mathcal{J}} \rangle$ such that $(d, 0) \in p_i^{\mathcal{J}}$ iff $I(p_i) = \mathit{false}$, and $(d, 1) \in p_i^{\mathcal{J}}$ otherwise. By construction, $(\mathcal{J}, d) \models C$, and $I \models S$ iff $(\mathcal{J}, d) \models \neg D$. Consequently, if S is satisfiable, then $C \sqsubseteq D$ is not valid.

Conversely, if $C \sqsubseteq D$ is not valid, then there exist $\mathcal{J}$ and $d \in \Delta^{\mathcal{J}}$ such that $(\mathcal{J}, d) \models C \sqcap \neg D$. Define a propositional interpretation I of S by setting $I(p) = \mathit{true}$ iff $(d, 1) \in p_i^{\mathcal{J}}$. By construction (and since d does not satisfy D), $I \models S$, which proves that if $C \sqsubseteq D$ is not valid, then S is satisfiable.

We conclude that the above reduction is correct. Moreover, it can be clearly computed in LOGSPACE. This proves that subsumption is coNP-hard even if the KB is empty.

Membership in coNP can be proved by showing that the complement of subsumption is in NP. Given a query $C \sqsubseteq D$, it suffices to choose nondeterministically one of the disjuncts C_i in the left hand side of the query, and replace each constraint $[\ell, u](f)$ occurring in C_i with a nondeterministically chosen disjunct from (9). Call C_i' the resulting concept and note that it is one of the disjuncts in C^*. Therefore, $\mathcal{K} \not\models C \sqsubseteq D$ iff for some nondeterministic choice, $\mathcal{K} \not\models C_i' \sqsubseteq D$. The latter subsumption test can be evaluated in deterministic polynomial time with STS, so the complement of $\mathcal{PL}$ subsumption is in NP. ∎

Acknowledgements

The use case about GDPR compliance illustrated with (5) and Example 2 is due to Benedict Whittam Smith (Thomson-Reuters). The author is grateful to the anonymous reviewers for their constructive comments. This research is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N. 731601.

References

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI-05*, pages 364–369. Professional Book Center, 2005.

[Baader *et al.*, 2008] Franz Baader, Carsten Lutz, and Sebastian Brandt. Pushing the EL envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[Bonatti *et al.*, 2010] Piero A. Bonatti, Juri Luca De Coi, Daniel Olmedilla, and Luigi Sauro. A rule-based trust negotiation system. *IEEE Trans. Knowl. Data Eng.*, 22(11):1507–1520, 2010.

[Borgida and Patel-Schneider, 1994] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. Artif. Intell. Res.*, 1:277–308, 1994.

[Haase and Lutz, 2008] Christoph Haase and Carsten Lutz. Complexity of subsumption in the $\mathcal{EL}$ family of description logics: Acyclic and cyclic tboxes. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *ECAI 2008 - 18th European Conference on Artificial Intelligence*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 25–29. IOS Press, 2008.

[Kagal *et al.*, 2003] Lalana Kagal, Timothy W. Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 63–. IEEE Computer Society, 2003.

[Kazakov *et al.*, 2013] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *Journal of Automated Reasoning*, 53:1–61, 2013.

[Kolovski *et al.*, 2007] V. Kolovski, J. Hendler, and B. Parsia. Analyzing web access control policies. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 677–686,, 2007. ACM.

[Uszok *et al.*, 2003] Andrzej Uszok, Jeffrey M. Bradshaw, Renia Jeffers, Niranjan Suri, Patrick J. Hayes, Maggie R. Breedy, Larry Bunch, Matt Johnson, Shriniwas Kulkarni, and James Lott. KAoS policy and domain services: Towards a description-logic approach to policy representation, deconfliction, and enforcement. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 93–96, Lake Como, Italy, June 2003. IEEE Computer Society.