A Canonical Form for PROV Documents and its Application to Equality, Signature, and Validation — Supplementary Material —

Luc Moreau, University of Southampton

Supplementary material for the paper "A Canonical Form for PROV Documents and its Application to Equality, Signature, and Validation".

CCS Concepts: •**Information systems** → **Semantic web description languages; Data encoding and canonicalization;** •**Security and privacy** → **Digital signatures;** •**Theory of computation** → **Data provenance;**

A. INFERENCES OVER MERGEABLE TERMS

The following inferences have to be applied to Mergeable Terms.

INFERENCE A.1 (TYPE INFERENCE). *(See PCO50 (typing))*

(*1*) IF used$(\Theta_o, \Theta_a, \Theta_e, \Gamma)$ THEN activity$(\Theta_a, [\,])$ AND entity$(\Theta_e, [\,])$.
(*2*) IF wasGeneratedBy$(\Theta_o, \Theta_e, \Theta_a, \Gamma)$ THEN entity$(\Theta_e, [\,])$ AND activity$(\Theta_a, [\,])$.
(*3*) IF wasInformedBy$(\Theta_o, \Theta_a^2, \Theta_a^1, \Gamma)$ THEN activity$(\Theta_a^2, [\,])$ AND activity$(\Theta_a^1, [\,])$.
(*4*) IF wasStartedBy$(\Theta_o, \Theta_a^2, \Theta_e, \Theta_a^1, \Gamma)$ THEN activity$(\Theta_a^2, [\,])$ AND entity$(\Theta_e, [\,])$ AND activity$(\Theta_a^1, [\,])$.
(*5*) IF wasEndedBy$(\Theta_o, \Theta_a^2, \Theta_e, \Theta_a^1, \Gamma)$ THEN activity$(\Theta_a^2, [\,])$ AND entity$(\Theta_e, [\,])$ AND activity$(\Theta_a^1, [\,])$.
(*6*) IF wasInvalidatedBy$(\Theta_o, \Theta_e, \Theta_a, \Gamma)$ THEN activity$(\Theta_e, [\,])$ AND activity$(\Theta_a, [\,])$.
(*7*) IF wasDerivedFrom$(\Theta_o, \Theta_e^2, \Theta_e^1, \Theta_a, \Theta_g, \Theta_u, \Gamma)$ THEN entity$(\Theta_e^2, [\,])$ AND entity$(\Theta_e^1, [\,])$ AND activity$(\Theta_a, [\,])$.
(*8*) IF wasAttributedTo$(\Theta_o, \Theta_e, \Theta_{ag}, \Gamma)$ THEN entity$(\Theta_e, [\,])$ AND agent$(\Theta_{ag}, [\,])$.
(*9*) IF wasAssociatedWith$(\Theta_o, \Theta_a, \Theta_{ag}, \Theta_e, \Gamma)$ THEN activity$(\Theta_a, [\,])$ AND agent$(\Theta_{ag}, [\,])$.
(*10*) IF actedOnBehalfOf$(\Theta_o, \Theta_{ag}^2, \Theta_{ag}^1, \Theta_a, \Gamma)$ THEN agent$(\Theta_{ag}^2, [\,])$ AND activity$(\Theta_a, [\,])$ AND agent$(\Theta_{ag}^1, [\,])$.
(*11*) IF alternateOf$(\Theta_o, \Theta_e^2, \Theta_e^1, \Gamma)$ THEN entity$(\Theta_e^2, [\,])$ AND entity$(\Theta_e^1, [\,])$.
(*12*) IF specializationOf$(\Theta_o, \Theta_e^2, \Theta_e^1, \Gamma)$ THEN entity$(\Theta_e^2, [\,])$ AND entity$(\Theta_e^1, [\,])$.
(*13*) IF hadMember$(\Theta_o, \Theta_c, \Theta_e, \Gamma)$ THEN entity$(\Theta_x, [\,])$ AND entity$(\Theta_e, [\,])$.

INFERENCE A.2 (WASINFORMEDBY INFERENCE). *(See PCO6 (generation-use-communication-inference))*

— IF wasGeneratedBy$(\Theta_o^1, \Theta_e^1, \Theta_a^1, attr^1)$ AND used$(\Theta_o^2, \Theta_a^2, \Theta_e^1, attr^2)$ THEN wasInformedBy$(\emptyset, \Theta_a^2, \Theta_a^1, [\,])$.

INFERENCE A.3 (SPECIALIZATIONOF AND ALTERNATEOF). *(See PCO12 (revision-is-alternate-inference), PCO16 (alternate-reflexive), PCO17 (alternate-*

This work is funded in part by the EPSRC SOCIAM (EP/J017728/1) and ORCHID (EP/I011587/1) projects, the FP7 SmartSociety (600854) project, and the ESRC eBook (ES/K007246/1) project.
Author's addresses: Luc Moreau, Web and Internet Science Group, Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ.
 1533-5399/2017/00-ART0 $15.00
DOI: http://dx.doi.org/10.1145/3032990

transitive), PCO18 (alternate-symmetric), PCO19 (specialization-transitive), PCO20 (specialization-alternate-inference))

(*1*) IF entity(Θ_e, Γ) THEN alternateOf$(\emptyset, \Theta_e, \Theta_e, [\,])$.
(*2*) IF alternateOf$(\Theta_o, \Theta_e^2, \Theta_e^1, \Gamma)$ AND alternateOf$(\Theta'_o, \Theta_e^1, \Theta_e^0, \Gamma')$ THEN alternateOf$(\emptyset, \Theta_e^2, \Theta_e^0, \emptyset)$.
(*3*) IF alternateOf$(\Theta_o, \Theta_e^2, \Theta_e^1, \Gamma)$ THEN alternateOf$(\emptyset, \Theta_e^1, \Theta_e^2, \emptyset)$.
(*4*) IF specializationOf$(\Theta_o, \Theta_e^2, \Theta_e^1, \Gamma)$ AND specializationOf$(\Theta'_o, \Theta_e^1, \Theta_e^0, \Gamma')$ THEN specializationOf$(\emptyset, \Theta_e^2, \Theta_e^0, \emptyset)$.
(*5*) IF wasDerivedFrom$(\Theta_o, \Theta_e^2, \Theta_e^1, \Theta_a, \Theta_g, \Theta_u, \Gamma \cup$ [prov:type = `prov : Revision`]) THEN alternateOf$(\emptyset, \Theta_e^2, \Theta_e^1, [\,])$.

INFERENCE A.4 (INFLUENCE). *(See PCO15 (influence-inference))*

(*1*) IF used$(\Theta_o, \Theta_a, \Theta_e, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_a, \Theta_e, attr)$.
(*2*) IF wasGeneratedBy$(\Theta_o, \Theta_e, \Theta_a, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_e, \Theta_a, attr)$.
(*3*) IF wasInformedBy$(\Theta_o, \Theta_a^2, \Theta_a^1, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_a^2, \Theta_a^1, attr)$.
(*4*) IF wasStartedBy$(\Theta_o, \Theta_a^2, \Theta_e, \Theta_a^1, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_a^2, \Theta_e, attr)$.
(*5*) IF wasEndedBy$(\Theta_o, \Theta_a^2, \Theta_e, \Theta_a^1, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_a^2, \Theta_e, attr)$.
(*6*) IF wasInvalidatedBy$(\Theta_o, \Theta_e, \Theta_a, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_e, \Theta_a, attr)$.
(*7*) IF wasDerivedFrom$(\Theta_o, \Theta_e^2, \Theta_e^1, \Theta_a, \Theta_g, \Theta_u, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_e^2, \Theta_e^1, attr)$.
(*8*) IF wasAttributedTo$(\Theta_o, \Theta_e, \Theta_{ag}, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_e, \Theta_{ag}, attr)$.
(*9*) IF wasAssociatedWith$(\Theta_o, \Theta_a, \Theta_{ag}, \Theta_e, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_a, \Theta_{ag}, attr)$.
(*10*) IF actedOnBehalfOf$(\Theta_o, \Theta_{ag}^2, \Theta_{ag}^1, \Theta_a, \Gamma)$ THEN wasInfluencedBy$(\Theta_o, \Theta_{ag}^2, \Theta_{ag}^1, attr)$.

B. HANDLING BUNDLES

The function *fusion* was defined for sets of terms. However, PROV documents can also contain bundles, which are named sets of terms. Bundles are transformed to a canonical form individually, by transforming a bundle's set of terms into canonical form. Two observations need to be made. First, if a document contains two bundles with the same name, then the bundles should be merged before being tranformed to canonical form; this allows large bundles to be streamed into split bundles with the same name and be regrouped by the recipient in a coherent manner. Second, the name of a bundle is defined in the scope of that bundle; thus, it is the membership function computed from the terms of a bundle that is applied to the name of that bundle.

$$
\begin{array}{rcl}
fusion & : & \mathbb{S}et(M\_Bundle) \to \mathbb{S}et(M\_Bundle) \\
fusion(b^*) & = & \textbf{let } b'^* = values(mapValues(groupBy(b^*, id{\downarrow}(x)), \lambda l.reduce(l, merge))) \\
 & & \textbf{let } \Psi = index\_membership_k(terms{\downarrow}(b)) \\
 & & \textbf{in } map(b'^*, \lambda b'.fusion(b'))
\end{array}
$$

The *fusion* function on bundles is also defined as a fixed point.

$$
\begin{array}{rcl}
fusion & : & M\_Bundle \to PCF\_Bundle \\
fusion(b) & = & \textbf{let } b_1 = fusion_k(fusion_i(b)) \\
 & = & \textbf{in if } b_1 = b \textbf{ then } b \textbf{ else } fusion(b_1)
\end{array}
$$

The two auxiliary functions $fusion_i$ and $fusion_k$ access terms in bundles, index and merge them, rearrange them, and construct a bundle. The resulting bundle also has its names replaced by equivalents names. Further, in $fusion_{i,k}$ the bundle identifiers are used as initial values by $index\_membership_{i,k}$ to find the membership function.

$$
fusion_i \quad : \quad M\_Bundle \to M\_Bundle
$$

$$
\begin{aligned}
fusion_i(b) &= \textbf{let } \langle indx, \Psi \rangle = index\_membership_i(b.t^*, \{b.id\}) \\
&\quad \textbf{in } \mathsf{bundle}(fmap(b.id, \Psi), fmap(indx.values, \lambda t.\mathcal{R}_\Psi(t))) \\
fusion_k &: M\_Bundle \to M\_Bundle \\
fusion_k(b) &= \textbf{let } \langle indx, \Psi \rangle = index\_membership_k(b.t^*, \{b.id\}) \\
&\quad \textbf{in } \mathsf{bundle}(fmap(b.id, \Psi), fmap(indx.values, \lambda t.\mathcal{R}_\Psi(t)))
\end{aligned}
$$

C. XML SIGNATURE

An example of signature is displayed in Figure 1. The structure of `SignedInfo` includes the canonicalization algorithm, a signature algorithm, and one or more references. `CanonicalizationMethod` is a required element that specifies the canonicalization algorithm applied to the `SignedInfo` element prior to performing signature calculations. `SignatureMethod` is a required element that specifies the algorithm used for signature generation and validation. The `Reference` element specifies a digest algorithm and digest value, and optionally an identifier of the object being signed, the type of the object, and/or a list of transforms to be applied prior to digesting.

The `URI` attribute of a `Reference` element identifies a data object using a URI-Reference. Here, `#id12345` is a URL relative to the current document, referring to node with Id `#id12345` (see Figure 3). The optional `Transforms` element contains an ordered list of Transform elements; these describe how the signer obtained the data object that was digested. `DigestMethod` is a required element that identifies the digest algorithm to be applied to the signed object. `DigestValue` is an element that contains the encoded value of the digest encoded in base 64. The `SignatureValue` element contains the actual value of the digital signature encoded in base 64. `KeyInfo` is an optional element that enables the recipient(s) to obtain the key needed to validate the signature. `KeyInfo` may contain keys, names, certificates.

```
<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="... -e4eac4c4a855">
    <dsig:SignedInfo>
        <dsig:CanonicalizationMethod Algorithm=" ... /2006/12/xml-c14n11"/>
        <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <dsig:Reference URI="#id12345">
            <dsig:Transforms>
                <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </dsig:Transforms>
            <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue> ... </dsig:DigestValue>
        </dsig:Reference>
    </dsig:SignedInfo>
    <dsig:SignatureValue> ...</dsig:SignatureValue>
    <dsig:KeyInfo Id="G302427c5-7af5-49f4-8556-e50c6560e839">
        <dsig:X509Data>
           <dsig:X509Certificate> ... </dsig:X509Certificate>
        </dsig:X509Data>
    </dsig:KeyInfo>
</dsig:Signature>
```

Fig. 1. XML Signature for the Document appearing in the Paper's Figure 3

D. CONSTRAINTS TABLE

Table I lists all the definitions, contraints and inferences defined in PROV-CONSTRAINTS, and summarises how they are handled in a validator based on the description of Section 5.3. Each column refers to some functionality described in the

paper, and the associated section is provided for completeness, except for the column "dropped", which refers to constraints that are not required to determine validity.

Table I: Summary Table of Constraints

	dropped	*inference* (Section 4.3)	*inject* (Section 4.4.1)	$fusion_i$ (Section 4.4.4)	$fusion_k$ (Section 4.4.4)	validation (Section 5.3)
PCO01 (optional-identifiers)			✓			
PCO02 (optional-attributes)			✓			
PCO03 (definition-short-forms)			✓			
PCO04 (optional-placeholders)			✓			
PCO05 (communication-generation-use-inference)	✓					
PCO06 (generation-use-communication-inference)		✓				
PCO07 (entity-generation-invalidation-inference)	✓					
PCO08 (activity-start-end-inference)	✓					
PCO09 (wasStartedBy-inference)	✓					
PCO10 (wasEndedBy-inference)	✓					
PCO11 (derivation-generation-use-inference)	✓					
PCO12 (revision-is-alternate-inference)	✓					
PCO13 (attribution-inference)	✓					
PCO14 (delegation-inference)	✓					
PCO15 (influence-inference)		✓				
PCO16 (alternate-reflexive)		✓				
PCO17 (alternate-transitive)		✓				
PCO18 (alternate-symmetric)		✓				
PCO19 (specialization-transitive)		✓				
PCO20 (specialization-alternate-inference)		✓				
PCO21 (specialization-attributes-inference)		✓				
PCO22 (key-object)				✓		✓(5.1)
PCO23 (key-properties)				✓		✓(5.1)
PCO24 (unique-generation)					✓	✓(5.1)
PCO25 (unique-invalidation)					✓	✓(5.1)
PCO26 (unique-wasStartedBy)					✓	✓(5.1)
PCO27 (unique-wasEndedBy)					✓	✓(5.1)
PCO28 (unique-startTime)						✓(5.2)
PCO29 (unique-endTime)						✓(5.2)
PCO30 (start-precedes-end)						✓
PCO31 (start-start-ordering)						✓
PCO32 (end-end-ordering)						✓
PCO33 (usage-within-activity)						✓
PCO34 (generation-within-activity)						✓
PCO35 (wasInformedBy-ordering)						✓
PCO36 (generation-precedes-invalidation)						✓
PCO37 (generation-precedes-usage)						✓
PCO38 (usage-precedes-invalidation)						✓
PCO39 (generation-generation-ordering)						✓
PCO40 (invalidation-invalidation-ordering)						✓
PCO41 (derivation-usage-generation-ordering)						✓
PCO42 (derivation-generation-generation-ordering)						✓
PCO43 (wasStartedBy-ordering)						✓
PCO44 (wasEndedBy-ordering)						✓
PCO45 (specialization-generation-ordering)						✓
PCO46 (specialization-invalidation-ordering)						✓
PCO47 (wasAssociatedWith-ordering)						✓
PCO48 (wasAttributedTo-ordering)						✓

Table I Continued:

PCO49 (actedOnBehalfOf-ordering)						✓
PCO50 (typing)		✓				✓
PCO51 (impossible-unspecified-derivation-generation-use)						✓
PCO52 (impossible-specialization-reflexive)						✓
PCO53 (impossible-property-overlap)						✓
PCO54 (impossible-object-property-overlap)						✓
PCO55 (entity-activity-disjoint)						✓(5.3)
PCO56 (membership-empty-collection)						✓