PTCRISync Specification

An ORCID-based Synchronization Framework for PTCRIS

Version 0.3
May 2015
About PTCRIS

The PTCRIS (Portuguese Current Research Information System) program from the FCT|FCCN (Fundação para a Computação Científica Nacional of the Portuguese Foundation for Science and Technology) aims to ensure the creation and sustained development of a national integrated information ecosystem, to support research management according to the best international standards and practices. This report specifies a synchronization framework (PTCRISync) that relies on ORCID as a central hub for information exchange between the various national systems and international systems. PTCRISync will enable researchers to register a given research output at one of the interconnected national systems, and automatically propagate that output to the remaining ones, thus ensuring global consistency of the stored information. For more information about PTCRIS, see http://ptcris.pt.

Who should read this document

This document is aimed at PTCRIS service managers and developers who wish to use ORCID as a central hub for synchronization. The document may also prove useful to the users of the services who wish to understand in detail how their records are being synchronized among the services.

Conformity to ORCID

The current version of this specification (0.3) is based on version 1.2 of the ORCID API.

Authors

This PTCRISync specification was commissioned by the FCT|FCCN to the High-assurance Software Laboratory (HASLab), from INESC TEC and Universidade do Minho, Portugal, and developed by:

- Alcino Cunha
- Nuno Macedo

Acknowledgments

This specification benefited a lot from the input and suggestions of the following people, to whom the authors would like to express their appreciation:

- João Mendes Moreira (FCT|FCCN, Portugal)
- Carlos Sousa Pinto (DSI, Universidade do Minho, Portugal)
- Luís Valério, Pedro Lopes (DeGóis, Portugal)
- José Carvalho (SDUM, Universidade do Minho, Portugal)
- António Lopes (ISCTE, Portugal)
- Andrea Bollini (CINECA, Italy)

License

The PTCRISync specification by the HASLab is licensed under a Creative Commons Attribution 4.0 International License. Based on work at http://ptcris.pt.
1 Introduction

PTCRIS (Portuguese Current Research Information System) is a program, officially initiated in May 2014, which aims to ensure the creation and sustained development of a national integrated information ecosystem, to support research management according to the best international standards and practices.

One of PTCRIS’ goals is to reduce the burden of research output management, by adopting an “input once, re-use often” principle. In order to achieve this goal, a synchronization framework will be developed that relies on ORCID\(^1\) – a community-based service that aims to provide a registry of unique researcher identifiers and a method of linking research outputs to these identifiers, based on data collected from external sources – as a central hub for information exchange between the various national systems (including CV management systems such as DeGôis, Open Access repositories such as RCAAP or SARI, and local CRIS systems) and international systems (WoK, Scopus, Datacite, etc). Figure 1 presents an overview of this architecture, with various PTCRIS services shown in orange and international systems in blue. Among other features, this framework will enable researchers (or managers) to register once a given research output at one of the interconnected national systems, and automatically propagate that output to the remaining ones, thus ensuring global consistency of the stored information.

The goal of this report is to specify such synchronization framework according to well-established principles of rigorous software engineering. We will start by specifying (an abstract view of) the data model of ORCID and of an archetypal example of a PTCRIS service (Section 2). Then, we will specify the desired notions of consistency between ORCID and such PTCRIS service (Section 3). The synchronization procedures that can be used to enforce such consistency will then also be specified (Section 4). The goal was to design these procedures such that they satisfy several “well-behavedness” properties, such as correctness or stability. In Appendix C we present a formal specification of both the data models and consistency notions. As natural language is inherently ambiguous, this formal specification can ultimately be used to clarify any doubts about the specification. However, we believe that, for the most part, the specification should be clear enough, and understanding this formal specification will not be necessary for most readers. The formal specification enabled us to verify the above properties and to automatically derive interesting scenarios (Section 5), that can be used both to clarify the consistency notion and synchronization procedures with all stakeholders, and to be used as test cases to verify the conformity of the implementation.

2 Data Models

The main difficulties in the design of this framework stemmed from fundamental differences between the data model of ORCID and that of most PTCRIS services. As such, before presenting the desired consistency notion and synchronization procedures, we briefly present such data models. Here, we present only a very abstract view of the information stored in such profiles, focusing only on research outputs, namely works, and only on the attributes that are relevant for their synchronization. Moreover, we only present a very abstract view of the information presented in such profile, focusing on the attributes that are relevant for the synchronization framework. An ORCID user profile contains additional information, which PTCRIS is also interested in synchronizing among its services. However, some of this information is trivial to synchronize (e.g., education affiliations) while other, albeit not trivial, may be synchronized following the technique presented in this specification for works (e.g., funding information).

The current version of this report is based on the ORCID API v1.2. Version 2.0, currently in early stages of development, is expected to affect some details of the described synchronization framework.

\(^1\)http://orcid.org/
but not the overall concepts. Although we are not aware of its expected release date, changes will be pointed out throughout the presentation when deemed relevant.

The specified synchronization framework operates at the user profile level, that is, it intends to synchronize user profiles from different PTCRIS services with the corresponding user profile from ORCID. The matching of users across these systems is outside the scope of this specification, but it is usually a trivial issue, since PTCRIS services can (and most already do) store the ORCID iD of the researcher in his profile. As such, the specification of the data models focuses only on a single user profile.

2.1 ORCID

An ORCID user profile (whose data model is to be denoted just by ORCID) consists of a set of works. For the effects of this specification, we will assume that each work is a record with the following information:

- **putcode**: The internal identifier of the work.
- **uids**: A (possibly empty) set of external unique identifiers (UIDs) of the work. Among the UIDs currently supported by ORCID we have, for example, the DOI, EID or ISBN.
- **source**: The source of the information in the work. The source can be the user itself or any other external source associated with ORCID, such as Scopus, CrossRef, or one of the PTCRIS services.
- **metadata**: Any meta-data associated with the work, such as its title, publication year, publication type, authors, etc.
- **preferred**: A boolean attribute marking whether this work is the one preferred by the user among similar ones (the definition of “similar” is presented below).

In this data model, the meta-data of a work is not further discriminated because the consistency relation and synchronization procedures (specified in the next sections) are, in general, oblivious to it and focus mainly on harvesting UIDs from ORCID profiles, being the meta-data handled in an opaque way. Note that the preferred boolean attribute is not directly returned by the ORCID API, but can
be inferred from the order in which the works are stored in an ORCID profile\(^2\). A UML class diagram depicting this data model is presented in Figure 2. Classes Putcode, UID, and Metadata are depicted as *primitive*, since that, for the effect of this abstract specification of the data model, they can be seen as atomic data values, with no further structure.

A distinctive feature of ORCID is the possibility of using different external sources to automatically populate a user profile. This means that a user profile can contain different works that actually describe the same research output (possibly containing different or even contradictory meta-data). The ORCID web interface already groups together works that describe the same output, showing only the preferred one in the overview. The grouping mechanism is quite simple, and just assumes two works \(w_1\) and \(w_2\) are *similar* if, and only if, they have a shared UID or there is another work \(w_3\) that is similar to \(w_1\) and similar to \(w_2\). Essentially, this recursive definition considers two works to be similar if, and only if, they share directly or indirectly (via transitivity) some UID.

This data model is subject to several structural constraints that are imposed by the ORCID system. In an ORCID user profile:

**ORCID1** The putcode uniquely identifies a Work.

**ORCID2** There cannot be two works with the same source with shared uids.

**ORCID3** Among sets of similar works exactly one of them is the preferred one.

In fact, it is possible to create two works with shared identifiers using the web interface (whose source is the user). However, as discussed with the ORCID team, in the future constraint ORCID2 will be more strictly enforced and is already enforced by the API: if an external source publishes a work with a UID that is shared with a previous work of the same source, only one of them will remain in the user profile. As such, the synchronization framework will be specified taking this constraint into account.

The goal of ORCID is that most of the works in its database have at least one UID associated. The ORCID API also forces every work from an external source to have some UIDs assigned, but

\(^2\)Version 2.0 of the ORCID API is expected to have the groups of similar works, as well as the preferred among them, explicitly defined in the ORCID data model.
works added by the user via the web interface may still have an empty set of UIDs. Our consistency framework will take this guideline into account.

Figure 3 depicts possible ORCID profiles, following the notation from Table 1. The profile at Figure 3a is a valid one, where all the defined constraints hold, with two groups of similar works. Note how similar works are defined by the shared UIDs and not by the associated meta-data. Also, even though two works from the same source must have disjoint UIDs (Work2 and Work3), they may be grouped together by a third similar work (Work4). Figure 3b shows instead an invalid profile, where all the constraints are broken: the putcode keys Putcode0 and Putcode1 are shared (ORCID1), Work2 and Work3 are from the same source and have overlapping UIDs (ORCID2) and no group has a work set as preferred (ORCID3). The proposed synchronization framework assumes that such invalid profiles never occur, and expects all the above constraints to be enforced by the ORCID service.

The ability of the synchronization framework to modify user profiles is limited by the functionalities provided by the ORCID API. The fundamental limitation is that a service may only affect works that belong to it, meaning that a PTCRIS service may only manage works whose source is itself. The service is able to freely create and delete works (as long as the UIDs do not overlap, as already exposed). Although not currently supported by the API, ORCID is expected to allow the direct modification of existing works. The synchronizer will not be able to directly modify the set of preferred works selected by the user, although that may happen indirectly if a preferred work is deleted from the ORCID profile, or if a new work unifies two groups of similar works, which may only have one preferred work. In these cases, it is not clear how the new preferred work of the group is selected by the ORCID service. Since the ORCID API does not yet support the update of existing works, keeping a work up-to-date requires the deletion and subsequent re-insertion of the work, which may impact the user selection of the preferred works.

2.2 Archetypal PTCRIS Service

The biggest difference of a user profile in a PTCRIS service (depicted in Figure 4) is that it does not support multiple versions of the same research output, nor the grouping feature of similar versions likewise to ORCID. Typically, the profile of a user in a PTCRIS service (whose data model is to be denoted just by PTCRIS) consists of a set of productions. For the purposes of this report, we will assume that each Production is a record with the following information:

key The internal identifier of the Production.
uids A (possibly empty) set of UIDs of the Production. We assume that the types of UIDs supported by the PTCRIS service are the same as ORCID.

metadata Any meta-data associated with the Production, such as its title, publication year, publication type, authors, etc.

exported A boolean field indicating whether the Production is selected to be exported to ORCID.

Although, for the effects of this specification, we assume that the shape of the meta-data supported by ORCID is the same as the one supported by a PTCRIS service, in reality that may not be the case. In fact, the types of supported meta-data can differ substantially and it is may not be trivial to convert one format to the other. However, as will be shown below, the proposed synchronization procedure is oblivious to the meta-data information and relies only on UIDs to match research outputs. Thus, no specific conversion procedure from one format to the other will be imposed, leaving to each PTCRIS service the decision of how to do so (at least in this version of the specification). As such, in the following specifications and scenarios whenever we mention that the meta-data of a given production in the PTCRIS profile should be the same as the meta-data of a work at ORCID, we are not enforcing strict equality but just requiring that they should as similar as possible, taking into account the differences in the supported types. Each PTCRIS service implementing this specification should clarify as precisely as possible how such conversion is made.

As will be presented in the next section, the synchronization framework is semi-automatic and notification based. As such, each service will be required to support two kinds of notifications in a user profile: creation notifications, to alert the user that a new production has been found in ORCID; and modification notifications, to alert the user that new UIDs for an existing production have been found. The latter will be particularly useful for propagating UIDs between different PTCRIS services, in particular from open access repositories that provide handles for research outputs to academic CV management services, such as DeGóis. A Notification is a record with the following information:

key The identifier of the Notification inside the service. The set of possible identifiers of notifications is shared with the set of possible identifiers of productions.

uids A (possibly empty) set of UIDs of the Production.

Additionally, a Creation notification contains the following attribute:

metadata Any meta-data associated with the new production, such as its title, publication year, publication type, authors, etc.

If the same notification system is to be used for other purposes, the data model must be adapted accordingly.

Figure 4 presents a UML class diagram depicting the PTCRIS data model. This data model is subject to several structural constraints that should be internally enforced by each PTCRIS service. In a PTCRIS user profile:
PTCRIS1 The key attribute uniquely identifies a Production.

PTCRIS2 If a Production is selected to be exported then it must have at least one uid.

PTCRIS3 If two productions at least one uid then at most one of them can be selected to be exported.

PTCRIS4 The key attribute uniquely identifies a Creation notification.

PTCRIS5 The key of a Creation notification must not be the key of an existing Production.

PTCRIS6 The key of a Modification notification must be the key of an existing Production.

PTCRIS7 In a Modification notification the set of uids must be non empty.

Constraint PTCRIS2 is enforced to follow the ORCID guidelines to avoid works in the user profile without UIDs. Constraint PTCRIS3 is a direct consequence of constraint ORCID2: since ORCID does not allow two works of the same source to share UIDs, the PTCRIS service must enforce that exported productions never share UIDs. Constraints PTCRIS5 and PTCRIS6 define the overall notification mechanism: the keys of modification notifications point to the production that is to be modified, while the keys of the creation notifications are unique, as they result in new productions if accepted. Constraint PTCRIS7 reflects the role of modification of proposing new UIDs.

Notice that maintaining these constraints, in particular PTCRIS2 and PTCRIS3, is not a trivial affair: for example, when accepting a notification to add a UID these constrains may be broken, because two productions that previously shared no UIDs may now share some. There are several valid alternatives for enforcing this constraint: for example, ask the user which of the productions should no longer be exported, or simply mark all conflicting productions as no longer exported.

Figure 5 depicting possible PTCRIS profiles, following the notation from Table 2. Figure 5a depicts a valid PTCRIS profile with two productions, one of which is set to be exported. There is also a creation notification and a modification notification, set to modify Production0. In contrast, Figure 5b depicts an invalid instance, since the two productions with shared unique identifiers are set to be exported (PTCRIS3), the key of the creation notification is not unique (PTCRIS5), the modification does not refer to an existing production (PTCRIS6) and does not have any UIDs assigned (PTCRIS7).

In the context of the presented data model, the role of the synchronization framework is solely to manage the notifications in the user’s PTCRIS profile, and not affect the actual productions. The
synchronizer is also not allowed to modify the set of productions set to be exported. This semi-
automatic approach provides the user with valuable information while still allowing him to control the
updates that are effectively applied to the profile.

3 Consistency Relations

The synchronization framework will be specified following well-established principles in the research
field of consistency management and repair. In particular, we will first specify when, for a given user,
his profile in ORCID is consistent with his profile in the PTCRIS service. Formally this consistency is
a predicate that given the two profiles returns a boolean indicating whether they are consistent or not,
that is a function with type $\text{ORCID} \times \text{PTCRIS} \rightarrow \text{Bool}$. Typically, this consistency predicate is specified
as a set of logical rules that must all be satisfied for the two profiles to be considered consistent.
Alternatively, this predicate can be seen as a relation that, for each ORCID user profile returns all
possible consistent PTCRIS user profiles (and vice-versa), hence the term consistency relation.

The consistency relation between an ORCID profile and a profile from a PTCRIS service will be
factorized in two modular consistency relations:

**IMPORTED**: $\text{ORCID} \times \text{PTCRIS} \rightarrow \text{Bool}$ This consistency relation should be enforced by every PTCRIS
service that wishes to rely on the synchronization framework to harvest new research outputs
from ORCID, namely new publications and new UIDs of known publications. The general prin-
ciple of IMPORTED is that every UID in a ORCID profile should be harvested. The enforcement
of this consistency relation should be semi-automatic, based on a notification system, giving
freedom to the user to select which outputs or UIDs he wishes to add to his PTCRIS profile.

**EXPORTED**: $\text{ORCID} \times \text{PTCRIS} \rightarrow \text{Bool}$ This consistency relation should be enforced by every PTCRIS
service that wishes to be an ORCID source, and export its productions to ORCID, ensuring
that other PTCRIS services can harvest them. The general principle of EXPORTED is that every
production selected to be exported in the PTCRIS profile should be inserted as a new work in
the ORCID profile and then automatically kept up-to-date.

These consistency relations are logically independent, in the sense that each can either hold or not,
independently of the value of the other. A PTCRIS service may also wish to implement the conjunction
of both, leading to a consistency relation we will denote as \text{SYNCED}:

\[
\text{SYNCED} : \text{ORCID} \times \text{PTCRIS} \rightarrow \text{Bool} \\
\text{SYNCED}(o, p) = \text{IMPORTED}(o, p) \land \text{EXPORTED}(o, p)
\]

3.1 IMPORTED Consistency Relation

As discussed above, the general principles of this consistency relation is that it should be notification
based, and that the PTCRIS service should harvest all UIDs present in the ORCID profile. Whenever
a new research output is found (one with previously unknown UIDs) the user should be notified by
a creation notification, and whenever new UIDs for an existing research output are found the user
should be notified by a modification notification that lists all the harvested UIDs of the output.

Since the PTCRIS services do not support grouping likewise to ORCID, some care must be made
to avoid the proliferation of productions and notifications that actually describe the same research
output. In particular, when an ORCID group of works is unknown to the PTCRIS service, the existence
of a single creation notification in the user profile, grouping all UIDs of similar works should suffice to
ensure consistency. Of course, this raises the issue of how the meta-data of such creation notification
should be filled in, as the meta-data of similar works can (and often does) differ. On first glance,
the obvious choice would be to pick the meta-data of the work selected as preferred by the user.
Unfortunately, the following reasons prevent us from enforcing that behavior:
Since all groups of similar works must have a work selected as preferred (essentially the one chosen to be displayed in the user web page), a default preferred is always chosen by ORCID when a new research output is imported or the current preferred one is deleted by the user.

The ORCID API does not currently distinguish default preferred works from user-selected ones.

This means that the user might not have sanctioned the meta-data present in his preferred works at the time they are being imported by the PTCRIS service.

Unfortunately, meta-data is of highly variable quality in ORCID, with some external sources currently publishing meta-data with gross mistakes, for example, wrong publication types.

As such, the proposed specification leaves some freedom for the PTCRIS services to choose how to fill in the meta-data of a creation notification. This behavior is assumed to be encapsulated in a procedure

\[ \text{extract} : P(W) \rightarrow \text{Metadata} \]

that somehow extracts the meta-data from a group of similar works. There are several sensible behaviors for this procedure, for example:

- Even taking into account the above reasons, just select the meta-data of the preferred work. This will be the behavior assumed in the scenarios presented in Section 5 and in the formal specification of the consistency relation. In this case, the service should clearly warn the user to actively verify his ORCID profile and choose as preferred works the ones with best meta-data (or create works with the user himself as source in case no work in a group has good meta-data).

- Implement an alternative mechanism for selecting the favorite work among a group of similar ones. For example, a service can allow the user to rank sources according to the quality of the provided meta-data, and then just select the meta-data from the work of the highest ranked source. Notice however, that even with constraint ORCID2 there can be two works of the same source that are similar (due to transitivity), so some additional (deterministic) mechanism must be implemented to further disambiguate the selection (for example, the most recent one).

- Ignore altogether the meta-data of the works in the ORCID profile and try to fetch meta-data from authoritative sources based on the UIDs of the group of similar works.

A PTCRIS service implementing this specification should clarify as precisely as possible how \text{extract} is implemented, so that the process is transparent to the end user.

Taking this into account, the constraints that must be satisfied in order for \text{IMPORTED} to hold are:

\text{IMPORTED1} For every Work in ORCID there exists an artifact in PTCRIS (either a Production or a Notification) that contains all uids of its similar works.

\text{IMPORTED2} Every Notification contains exactly the uids of a group of similar works.

\text{IMPORTED3} The metadata of every Creation notification is computed using \text{extract} over the group of similar works from which its uids were aggregated.

\text{IMPORTED4} Every Creation notification must have a non-empty set of uids that are not shared with any Production nor any other Notification.

\text{IMPORTED5} Every Modification notification must share some (but not all) uids with the associated Production (the one with the same key).

\text{IMPORTED6} For every group of similar works, every Production that shares some (but not all) of its uids must be affected by exactly one Modification notification to introduce the group's uids.
The first constraint, IMPORTED1, is more or less a direct translation of the general principle that all UIDs must be harvested. Constraint IMPORTED2 forbids the existence of notifications not harvested from ORCID (we assume that the notification system is only used for synchronization with ORCID – if the notification system is used for other purposes then this constraint no longer makes sense). Constraint IMPORTED3 regards the extraction of the meta-data from a group of similar works for creation notifications, as mentioned above.

Constraint IMPORTED4 forbids the existence of creation notifications when a production for the same output already exists (a modification notification should be used instead in those cases), as well as duplicated creation notifications. It also forbids creation notifications without UIDs. The reason for this is that, since our research output matching procedure is based just on UIDs instead of metadata, there is no way to determine precisely whether a work without UIDs is already in the PTCRIS profile. If the creation of such works was notified and then later accepted by the user, undesirable notifications for the same works would reappear. The choice to implement a matching procedure based just on UIDs is mainly to preserve uniformity in the design, more specifically, to use the same matching procedure that ORCID currently uses: we believe that having a different matching procedure in the PTCRIS synchronization framework would be confusing for the end user, and should be avoided.

Constraint IMPORTED5 restricts modification notifications to be applied only to an existing production that shares UIDs with the harvested output. Moreover, it also prevents spurious modification notifications that do not add any new UIDs to the associated production. Notice that, to simplify the presentation, we assume that a modification notification contains all the UIDs that should be present in the associated production, and not only the new ones. Of course, from a user interface perspective it would be convenient to show only the new ones, that can easily be computed by set difference. Finally, IMPORTED6 forces exactly one modification notification for every production that matches a group of similar works in ORCID, but still misses some UIDs of that group.

It is relevant to note that these constraints are independent, meaning that none of the constraints is implied by the others. They are also not implied by the constraints imposed over the data models. In contrast, there are actually constraints over the data model that are implied by constraint IMPORTED5 (forcing modification notifications to share UIDs with the referred production): whenever this constraint holds, then PTCRIS6 (key points to existing production) and PTCRIS7 (some UIDs in the notification) also hold.

3.2 EXPORTED Consistency Relation

This consistency relation is considerably simpler than the previous one. As mentioned above, the general principle of EXPORTED is that every production marked as exported in a PTCRIS profile should be inserted as a new work in the ORCID profile and then automatically kept up-to-date. As such, for EXPORTED to hold, the following constraints must be satisfied:

EXPORTED1 For every exported Production exactly one Work must exist in ORCID with the same uids, the same metadata, and whose source is the PTCRIS service.

EXPORTED2 Dually, for every Work whose source is the PTCRIS service, a Production must exist in PTCRIS with the same uids, the same metadata, and that is selected as exported.

Essentially these two dual constraints, together with the data model constraints (namely ORCID2 and PTCRIS3), ensure that there must exist a one-to-one correspondence between productions selected to be exported and works whose source is the PTCRIS service.

Again these constraints are independent and are not implied by the constraints imposed over the data models. None of the data models constraints are entailed by this consistency relation.
4  Synchronization Procedures

When the user profiles at ORCID and at a PTCRIS service are inconsistent, a synchronization procedure must be run in order to recover such consistency. In this section we propose two separate synchronization procedures to be used when the service intends to enforce consistency according to IMPORTED or EXPORTED, respectively. These modular synchronization procedures can be combined in a precise way, to recover the consistency in services that are enforcing both consistency relations, i.e. SYNCED.

IMPORT:  ORCID × PTCRIS → PTCRIS  This synchronization procedure should be used to enforce the IMPORTED consistency relation. The main principle is that it does not change the ORCID user profile. Moreover the only changes it produces to the PTCRIS profile are the addition and deletion of notifications.

EXPORT:  ORCID × PTCRIS → ORCID  This synchronization procedure should be used when enforcing the EXPORTED consistency relation. The main principle is that it does not change the PTCRIS user profile. Moreover the only changes it produces to the ORCID profile are the addition / deletion / modification of works whose source is PTCRIS.

These synchronization procedures were designed to satisfy several “well-behavedness” properties and thus achieve a trustworthy design. These properties are described in Appendix B.

The two specified synchronization procedures can be combined in the following way to obtain a synchronizer to enforce SYNCED, the full consistency of the user profiles according to both IMPORTED and EXPORTED (to be used by services that wish to enforce both):

\[
\text{SYNC}: \text{ORCID} \times \text{PTCRIS} \rightarrow \text{ORCID} \times \text{PTCRIS} \\
\text{SYNC}(o, p) \triangleq \text{let } o' = \text{EXPORT}(o, p) \text{ in } (o', \text{IMPORT}(o', p))
\]

Each service is free to choose when to run these synchronization procedures, as long as inconsistencies in the profiles are eventually resolved within a reasonable delay. One possible choice would be run them periodically in the specified order in batch mode, thus avoiding possible delays that can negatively affect the user interface. Another sensible choice would be to run IMPORT at the begin of a user session and EXPORT at the end. This ensures that the visible parts of the profiles are consistent when the user is logged out, but that whenever he logs in again the correct notifications are shown. We think that invoking the synchronization procedures every time the user performs an edit within a session may be counterproductive, as new notifications might keep popping-in and confuse the user. Similarly to distributed systems, the goal of the synchronization framework is to ensure eventual consistency and not necessarily real-time strong consistency among all services.

4.1  IMPORT Procedure

The IMPORT procedure can be implemented with the following broad steps:

1. Delete all existing notifications;

2. For every group of similar works in ORCID, if none of their uids is shared by a Production in PTCRIS, create a Creation notification with all the uids of the group, and using extract to compute the metadata.

3. For every group of similar works in ORCID, if some of their uids (but not all) are shared by a Production in PTCRIS, create a Modification notification associated with that Production (with the same key) with all the uids of the group.
**IMPORT**(*o*: ORCID, *p*: PTCRIS) ≜

1:  *p*.notifications ← ∅
2:  *ws* ← \{*x* | *x* ∈ *o*.works ∧ *x*.uids ≠ ∅\}
3:  while *ws* ≠ ∅
4:      choose *w* ∈ *ws*
5:         *g* ← similar(*o*, *w*)
6:         *us* ← \{*x*.uids | *x* ∈ *g*\}
7:         *ps* ← \{*x* | *x* ∈ *p*.productions ∧ *x*.uids ∩ *us* ≠ ∅\}
8:         if *ps* = ∅
9:             *n* ← new Creation(*us*, extract(*g*))
10:            *o*.notifications ← *o*.notifications ∪ \{*n*\}
11:       else
12:            while *ps* ≠ ∅
13:                choose *x* ∈ *ps*
14:                    if *us* ∉ *x*.uids
15:                        *n* ← new Modification(*x*.key, *us*)
16:                        *o*.notifications ← *o*.notifications ∪ \{*n*\}
17:                *ps* ← *ps* \ {*x*}
18:            *ws* ← *ws* \ {*g*}

Figure 6: The IMPORT procedure.

The key point of this procedure is to iterate over all groups of similar works in the ORCID profile. Unfortunately, ORCID’s API does not identify such sets (as specified in our data model), and thus to implement this procedure one must first compute them. By seeing the user profile as an undirected graph whose nodes are works and where a edge between two works exists if, and only if, they have shared UIDs, given a work, its set of similar works is precisely the set of works that are reachable from it (determined by a simple traversal).

Figure 6 presents a possible algorithm that implements the IMPORT procedure. As described above, it starts by deleting all existing notifications (line 1) and computing the set *ws* of all works that should be considered for importing, namely those with some UIDs (line 2). While there are works to be processed, it picks one of them and stores it in variable *w* (line 4), and then computes its set *g* of similar works using the traversal procedure presented in Figure 7 (line 5) and the set *us* of all UIDs of the set *g*. Notice that, by definition of similar, *w* is also contained in *g*. The next step is to compute the set *ps* of all productions in the PTCRIS profile that share UIDs with *g* (line 7). If this set is empty then a new creation notification is added to the set of notifications (lines 9 and 10). Here we assume the existence of a constructor for creation notifications that, given a set of UIDs and the desired meta-data (to be computed with the extract procedure), returns a new creation with a fresh key. If the set *ps* is not empty, then for all productions in *ps* that do not contain all the UIDs in *us*, a new modification notification is added to insert the missing UIDs (lines 12 to 17). Here we assume the existence of a constructor for modification notifications that, given a key of an existing production and a set of UIDs, returns a new modification. An iteration of the outermost cycle ends by removing all similar works in the set *g* from the set *ws* of works that still need to be processed.
\[\text{similar}(o : \text{ORCID}, w : \text{Work}) \equiv \]

1: \[g \leftarrow \emptyset\]
2: \[f \leftarrow \{w\}\]
3: \[\text{while } f \neq \emptyset\]
4: \[\text{choose } x \in f\]
5: \[f \leftarrow f \setminus \{x\}\]
6: \[g \leftarrow g \cup \{x\}\]
7: \[a \leftarrow \{y \mid y \in o.\text{works} \land x.\text{uids} \cap y.\text{uids} \neq \emptyset\}\]
8: \[f \leftarrow f \cup (a \setminus g)\]
9: \[\text{return } g\]

Figure 7: The similar procedure.

The IMPORT procedure can be implemented quite efficiently. The most critical step that requires some care implementing is the one on line 7, determining the set of productions \(p_a\) that share some UIDs with \(x\). A naïve implementation of this step, just by iterating over all productions for every group, would lead to an approximate complexity \(O(|\text{works}| \times |\text{productions}|)\) (assuming there is an average constant number of UIDs per research output, independent of the number of works and productions present in the user profiles). This complexity can be improved by maintaining a reverse map in the PTCRIS service that, for each UID returns the productions that contain it: this would allow an implementation of the step in line 7 that quickly narrows the set of matching productions\(^3\).

If each group of similar works has an average constant number of productions that match it (again a reasonable assumption for real user profiles), this optimization would lower the complexity to the class \(O(|\text{works}|)\), that is, linear in the number of works currently in the ORCID user profile. Notice that, to attain this complexity, a similar optimization should be made in line 7 of procedure similar, which for a given work \(x\) computes the set \(a\) of adjacent works, that is, works with shared UIDs. Again, to avoid iterating over all works, when parsing the ORCID user profile a reverse mapping should be constructed that, for each UID, returns the set of works that contain it. This reverse mapping can be constructed in linear time in the number of works\(^4\).

The specified IMPORT synchronization procedure is, in fact, the only admissible one, given the constraints of the consistency relation specified in the previous section. The IMPORTED consistency relations is deterministic in the following sense: given two inconsistent profiles there is only one possible update to the PTCRIS profile that will render them consistent (affecting only the notifications and, again, modulo differences in the key attribute of Creation notifications).

### 4.2 EXPORT Procedure

On first glance, it seems that the EXPORT procedure could trivially be implemented by just deleting all works whose source is the PTCRIS service and then iterate over all productions selected to be exported and, for each, create a new matching work in the ORCID profile. This naïve procedure would ensure correctness, i.e., ensure that the resulting user profiles are consistent according to EXPORTED. However it would not be stable, since it could unnecessarily change the status of works selected as preferred (concretely, if works whose source is the PTCRIS service are set as preferred).

---

\(^3\)Notice that, in a standard relational database implementation, assuming that a table associating each Production Key with its uids exists, this reverse mapping can easily be implemented by maintaining an index on the UID column.

\(^4\)If, as expected, version 2.0 of the ORCID API explicitly supports groups of works in the ORCID data model, the IMPORT procedure will be simplified, since it will not no longer need to compute the groups of similar works.
As such, special caution must be exercised to ensure stability and, in general, minimize changes to the preferred status of existing works. Concretely, to avoid unnecessary changes in the preferred selection and preserve stability, the EXPORT synchronization procedure must at least guarantee that:

\textbf{EXPORTED3} Every preferred Work in the ORCID profile whose uids and metadata are those of an exported Production in the PTCRIS profile should be not modified.

Figure 8 presents a possible algorithm that implements such correct and stable procedure. It assumes that the ORCID API allows updates of existing works without changing their preferred status. It is not clear at the moment if that will indeed be the case. If not, then updating a work can be implemented by a deletion followed by the insertion the desired updated one, with the consequent potential loss of the preferred status. This scenario would still satisfy EXPORTED3, assuming that works that need not be updated are preserved.

The algorithm starts by computing the set $es$ of productions selected as exported in the PTCRIS profile (line 1) and the set $ws$ of works in the ORCID profile whose source is the PTCRIS service (line 2). Then, for each work $w \in ws$, it computes the set $ps \subseteq es$ of productions that share some UIDs with $w$ (line 5): if this set is empty, meaning that no matching exported production exists, the work $w$ is deleted from the ORCID profile (line 7); if it is non-empty, it contains at most one matching exported production $e$ (due to the data-model constraints), and the UIDs and meta-data of $w$ are updated to be consistent with those of $e$ (lines 10 and 11). If the update procedure in the ORCID API does not ensure the preservation of the preferred status, or if the update needs to be implemented by deletion followed by insertion, then the update in lines 10 and 11 should be conditional to $e$ having some difference to $w$, either in the UIDs or in the meta-data (to ensure stability). In the latter case, the production $e$ is also removed from the set $es$ containing the exported productions that still need to be processed.
After deleting or updating the previously existing works, the algorithm proceeds by creating a new work \( w \) with the PTCRIS service as source for each remaining exported production \( e \) (lines 14 to 18). Likewise to IMPORT, concerning efficiency, the most critical step of this procedure is line 5, and similar measures to those proposed above should be taken to ensure an efficient implementation.

The specified EXPORT synchronization procedure is, up to differences in preferred works and put-code identifiers, the only admissible one, given the constraints of the EXPORTED consistency relation specified in the previous section and the limitation to only modify works of which it is the source. Essentially, this consistency relation forces only one possible outcome for the content of works with source PTCRIS, but, due to the idiosyncrasies of the ORCID API already presented, it cannot always control which works will be the preferred ones. We assume however some sensible ORCID behavior in this situations. Namely, a work can only be the preferred of the group if it was already preferred previously (not necessarily of the same group – see Scenario 12) or the previous preferred of its group of similar works was deleted (meaning that the previous preferred had source PTCRIS and is no longer exported – see Scenario 10).

5 Synchronization Scenarios

This section presents some scenarios illustrating the specified synchronization procedures. A more exhaustive set of scenarios is presented in Appendix A. These scenarios were generated automatically from our formal specification presented at Appendix C.

A typical scenario involves a consistent state of the PTCRIS and ORCID profiles, a user update that may or not render the profiles inconsistent, and an execution of the synchronization procedure that recovers the consistency, if needed. Each scenario is followed by a graphical representation of the profiles. Tables 1 and 2 explain the notation that will be followed in this representation for the different artifacts.

5.1 IMPORT scenarios

The scenarios described in this section depict typical use cases of the synchronization framework when harvesting research outputs from ORCID.

Let us assume the most basic scenario where both the PTCRIS and the ORCID profiles of the user are empty (denoted by the \( \emptyset \) symbol), until a group of similar works from different external sources is introduced in the ORCID profile, resulting in the following configuration:

```
IMPORTED
EXPORTED
```

At this point, the synchronization procedure will detect a group of similar works in ORCID whose UIDs do not overlap with any of the PTCRIS productions, rendering the two profiles IMPORTED-inconsistent. To recover consistency, the IMPORT procedure will create a single creation notification with the aggregated UIDs of the group, and with the meta-data extracted from the preferred work (Scenario 1):
Eventually, the PTCRIS user accepts this creation notification, which results in a new production in his PTCRIS profile. This user update preserves the consistency between the two profiles:

Consider now that one of the external sources introduced a new work in the ORCID profile with an UID (DOI1) that is not yet known in the PTCRIS profile:

The UIDs of this work are unique among the works from that source (Scopus), but it is similar to the existing group by transitivity. The synchronization frameworks detects this similarity and proposes only a modification notification to introduce the new UIDs in the existing production, rather than a creation notification (Scenario 2):
The PTCRIS user eventually accepts the modification notification and introduces the new UIDs into the existing production, preserving the consistency between the profiles:

At some point, another similar work is introduced in the ORCID profile, whose UIDs already occur in the PTCRIS profile. This update does not break the consistency of the system, and the PTCRIS profile is not updated, preserving the stability of the system (Scenario 3):

Now, imagine that for some reason (maybe the user finds the meta-data incorrect, maybe he does not trust the source), the user deletes a work that connected all the works into a single group (Work1).
While there are now two distinct groups of similar works in the ORCID profile, since the production already contains every UID present in them, the profiles remain consistent (Scenario 4):

Yet, noticing this new disposition of groups, the user decides to manually remove the UID from the production that connected the two groups (DOI1), breaking the IMPORTED-consistency in the process:

The removed UID occurs in the ORCID profile but is no longer known to the PTCRIS profile. Moreover, there are now no shared UIDs between the production and the one of the groups (comprised solely by Work3), so the synchronization procedure adds a creation notification, rather than a modification notification (Scenario 5):

The PTCRIS user eventually accepts the creation notification and introduces a new production in his profile:

Then, perhaps by distraction (or because he allowed the external sources to automatically insert works), the work previously removed is reintroduced in the ORCID profile, connecting again all the works into a single group of similar works:
In this scenario however, both productions now share UIDs with the unified group of works, so a modification notification must be created for each production in order to recover imported-consistency (Scenario 6):

However, aware that this ORCID work is not to be trusted, the user chooses not to accept the notifications at the PTCRIS profile, and again removes the work from the ORCID profile:

The procedure should then remove the obsolete modification notifications from the PTCRIS profile in order to recover the IMPORTED-consistency (Scenario 7):
5.2 **EXPORT scenarios**

At this point, the user decided to clean up the meta-data (Metadata2) of a production on the PTCRIS side, and have it propagated to the ORCID profile. To do so, he sets the desired production to be exported:

To recover EXPORTED-consistency, the synchronization procedure must create a new work in the ORCID profile identical to the exported production (Scenario 8):

The EXPORT procedure has only the ability to update the ORCID profile, so, when a work that has the PTCRIS service as source is deleted by the user using the ORCID web interface:
The procedure will re-introduce it, since it is still selected to be exported in the PTCRIS profile (Scenario 9):

To effectively remove works from the ORCID profile inserted by the PTCRIS service, the user must deselect them as exported in the PTCRIS profile.

Since the user manually updated the meta-data from this production, he assumes that it is the best one, and sets the respective work as preferred in the ORCID profile. This does not affect the consistency of the profiles. If he now deselects the production from being exported:

The procedure will remove the work whose source was the PTCRIS service form the ORCID profile. Since this work was selected as the preferred one, it is not clear which work will be promoted to preferred by the ORCID system. One possibility would be (Scenario 10):

5.3 SYNC scenarios

In certain scenarios, fixing EXPORTED-consistency may lead to IMPORTED-inconsistencies, and thus the overall procedure SYNC should be applied. Consider that the user introduced the ambiguous UID
in the production and selected it to be exported:

As expected, EXPORT will recover the EXPORTED-consistency by creating a new identical work in the ORCID profile:

However, since this new work unifies the two pre-existing groups of the ORCID profile, the user should now be notified to introduce the remainder UIDs in the other production of the PTCRIS profile, so IMPORT must be subsequently executed (Scenario 12):
References


A Scenarios

Scenario 1
This scenario depicts the introduction of group of works in the ORCID profile without any productions with shared UIDs in the PTCRIS profile.

**Consistent profiles:** Empty PTCRIS and ORCID profiles.

**ORCID update** (⇒**IMPORTED-inconsistent**): A group of similar works (Work0 and Work1 through EID1) is introduced in the ORCID profile.

**IMPORT executed:** IMPORT introduces a creation notification in the PTCRIS profile and recovers IMPORTED-consistency.

**Observations:** The meta-data is assumed to have been extracted directly from the preferred work (Work0).
Scenario 2
This scenario depicts the introduction of new work in the ORCID profile that is similar to existing works but introduces new UIDs.

**Consistent profiles:** A production in PTCRIS profile matching a group of similar works in the ORCID profile.

**ORCID update (⇒IMPORTED-inconsistent):** A new work (Work2), similar through Handle1, is added in the ORCID profile, introducing a new UID (DOI0).

**IMPORT executed:** Since the existing Production0 in the PTCRIS profile shares UIDs with the group of the ORCID profile, IMPORT adds a modification notification over the production so that the new UID (DOI0) is introduced.

**Observations:** NA
**Scenario 3**
This scenario depicts the introduction of new work in the ORCID profile that is similar to existing works and does not introduce new UIDs.

**Consistent profiles:** A production in PTCRIS profile matching a group of similar works in the ORCID profile.

**ORCID update (⇒IMPORTED-inconsistent):** A new work (Work3), similar through DOI0, is added in the ORCID profile, without introducing new UIDs.

**IMPORT executed:** Since Production0 in the PTCRIS profile already contains all the UIDs of the group, the profiles were already consistent, so IMPORT does not perform any change.

**Observations:** NA
Scenario 4
This scenario depicts the deletion of a work in the ORCID profile that splits a group of similar works into two distinct works.

**Consistent profiles:** A production in the PTCRIS profile matching a group of similar works in the ORCID profile.

**ORCID update (⇒IMPORTED-inconsistent):** The work that rendered all works similar (Work1) is removed, splitting the group into two.

**IMPORT executed:** Since Production0 in the PTCRIS profile already contains all the UIDs of the two groups, the profiles are already consistent and IMPORT does not perform any change.

**Observations:** NA
Scenario 5

This scenario depicts the deletion of an UID in a production of the PTCRIS profile, so that not every group at the ORCID group is represented in the PTCRIS profile.

**Consistent profiles:** A production in the PTCRIS profile matching two distinct groups of similar works in the ORCID profile.

**PTCRIS update (⇒ IMPORTED-inconsistent):** An UID (DOI1) is removed from the production, so that it no longer matches with one of the groups in the ORCID profile (Work3).

**IMPORT executed:** Since the production does not share any UID with Work3, IMPORT creates a new creation notification in the PTCRIS profile.

**Observations:** NA
Scenario 6
This scenario depicts the introduction of an ORCID work that unifies previously distinct groups of similar works, now sharing UIDs with two productions at the PTCRIS profile.

**Consistent profiles:** Two groups of similar works whose UIDs match those of two productions at the PTCRIS profile.

**ORCID update (⇒IMPORTED-inconsistent):** A new work (Work2) is introduced in the ORCID profile that connects the two distinct groups of similar works.

**IMPORT executed:** Since the group of similar works now share UIDs with both productions at the PTCRIS profile, IMPORT creates a modification notification to introduce the missing UIDs for each of the productions.

**Observations:** NA
Scenario 7

This scenario depicts the removal of works from the ORCID profile that render some notifications in the PTCRIS profile obsolete.

**Consistent profiles:** A group of similar works in the ORCID profile shares UIDs with productions in the PTCRIS profile, to which modification notifications were assigned to introduce the additional UIDs.

**ORCID update (⇒ IMPORTED-Inconsistent):** A work was removed, dividing the group of similar works into two, that no longer match with both the productions of the PTCRIS profile.

**IMPORT executed:** Since each group only shares UIDs with one of the productions, the modification notifications are no longer valid, and must be removed by IMPORT.

**Observations:** NA
Scenario 8

This scenario depicts the selection of a production to be exported when some of its UIDs are shared by a group of similar works in the ORCID profile.

Consistent profiles: Two productions in the PTCRIS profile share UIDs with two groups of similar works in the ORCID profile.

PTCRIS update (⇒EXPORTED-inconsistent): One of the productions (Production0) is selected to be exported, requiring the creation of an identical work in the ORCID profile.

EXPORT executed: The EXPORT procedure creates the new work (Work4) with the PTCRIS service as the source, that is grouped with the existing works that share UIDs.

Observations: NA
Scenario 9
This scenario depicts the removal of a work from the ORCID profile whose source was a PTCRIS service.

**Consistent profiles:** A production from the PTCRIS profile is selected to be exported (Production0) and the matching work was created in the ORCID profile (Work4).

**ORCID update** (→ EXPORTED-inconsistent): The user removes the work whose source was the PTCRIS service from the ORCID profile (Work4).

**EXPORT executed:** Since the production selected to be exported no longer has a matching work at the ORCID profile, the work is re-introduced by EXPORT (Work4).

**Observations:** Reflects the design decision to not allow EXPORT to change the set of exported productions.
Scenario 10

This scenario depicts the deselection of an exported work when the matching work at the ORCID profile was selected as the preferred work.

Consistent profiles: A production from the PTCRIS profile is selected to be exported (Production0) and the matching work was created in the ORCID profile (Work4). This work was selected by the user as the preferred of the group of similar works.

PTCRIS update (⇒EXPORTED-inconsistent): The user deselects the production to be exported (Production0).

EXPORT executed: The work whose source was the PTCRIS service (Work4) no longer has a matched exported production in the PTCRIS profile, and thus must be deleted by EXPORT. Since it was the preferred of the group, one of the other similar works must be promoted.

Observations: The outcome of this scenario depends on the behavior of the ORCID service when a preferred work is deleted: while in this instance Work1 was promoted to be the preferred work of the group, the promotion of Work0 would be equally possible.
**Scenario 11**

This scenario depicts update of an exported work when the matching work at the ORCID profile was selected as the preferred work.

**Consistent profiles:** A production from the PTCRIS profile is selected to be exported (Production0) and the matching work was created in the ORCID profile (Work4). This work was selected by the user as the preferred of the group of similar works.

**PTCRIS update** (⇒ EXPORTED-inconsistent): The user updates the production to be exported (Production0 through Metadata3).

**EXPORT executed:** The work whose source was the PTCRIS service (Work4) is no longer identical to the exported production, and must be updated by EXPORT.

**Observations:** This scenario assumed that the update of works in ORCID is still not available. While this is still a consistent solution, it does affect the user selection of the preferred work. While in this instance Work1 was promoted to be the preferred work of the group, the promotion of Work0 would be equally valid, but not the preservation of Work4.
Scenario 12

This scenario depicts the exportation of a production that unifies two groups of works in the ORCID profile, leading to the need of further UIDs updates in the PTCRIS profile, and thus to the subsequent execution of the IMPORT procedure.

Consistent profiles: Two productions in the PTCRIS profile share UIDs with each other and with two groups of similar works in the ORCID profile.

PTCRIS update (⇒ EXPORTED-inconsistent): One of the productions (Production0) is selected to be exported, and since it shares UIDs with both groups of works will unify them under a single group of similar works.

SYNC executed: To recover full consistency, first EXPORT must create the work at the ORCID profile whose source is the PTCRIS service (Work5). This will unify the two groups, and thus the UIDs must be introduced in all matched productions in the PTCRIS profile (including Production1), which renders the profiles IMPORTED-inconsistent. The subsequent execution of IMPORT creates the modification notification.

Observations: The outcome of this scenario depends on the behavior of the ORCID service when a group is merged: while in this instance Work3 was preserved as the preferred (and Work1 demoted), the contrary would be equally possible.
B Trustworthy synchronization procedures

The procedures specified in Section 4 were designed to satisfy several “well-behavedness” properties. The most important of those is correctness, namely ensuring that after running the synchronization procedures the user profiles in ORCID and in the PTCRIS service are indeed consistent. Given \( o \), a user profile in ORCID, and \( p \), a user profile in PTCRIS, the following holds:

\[
\text{IMPORTED}(o, \text{IMPORT}(o, p)) \\
\text{EXPORTED}(\text{EXPORT}(o, p), p)
\]

Another important “well-behavedness” law is stability, ensuring that if we run the synchronization procedures on already consistent profiles the result is the same (modulo differences in the key identifier of creation notifications in the case of IMPORT – in the following law we use the comparison symbol \( \approx \) on PTCRIS profiles precisely to account for such possible differences\(^5\)):

\[
\text{IMPORTED}(o, p) \Rightarrow \text{IMPORT}(o, p) \approx p \\
\text{EXPORTED}(o, p) \Rightarrow \text{EXPORT}(o, p) = o
\]

These laws are the standard “well-behavedness” laws in synchronization frameworks, namely on frameworks for bidirectional transformation, whose goal is precisely to maintain two artifacts consistent by means of two transformations to propagate updates from each to the other (for an overview of this research field please see [1]). To be more precise, our formalization is based on the concrete framework of constraint maintainers, first proposed by Meertens [3], and later used by Stevens [5] to formalize the standard OMG bidirectional transformation language QVT-R [4].

Having stable synchronization procedures ensures that there is no need to explicitly check the consistency to determine if they should be run. If the user profiles in ORCID and in the PTCRIS service are consistent, running the specified synchronizers will not affect them. In fact, the checking procedures have the same approximate complexity as the synchronizers, and thus, no significant performance gains would be achieved by running them beforehand. In fact, we could even have a performance degradation if the user profiles happen to be inconsistent. These are the reasons why in Section 3 we did not specify a (non-declarative) procedure to explicitly check the consistency, as we see no utility for it in the synchronization framework if the specified synchronizers are correctly implemented.

As presented in Section 4, the two specified synchronization procedures can be combined in the following way to obtain a synchronizer to enforce SYNCED:

\[
\text{SYNC} : \text{ORCID} \times \text{PTCRIS} \rightarrow \text{ORCID} \times \text{PTCRIS} \\
\text{SYNC}(o, p) \triangleq \text{let } o' = \text{EXPORT}(o, p) \text{ in } (o', \text{IMPORT}(o', p))
\]

The specified order of execution is not arbitrary. In fact, it is the only order that ensures that the resulting procedure is both correct and stable:

\[
\text{SYNCED}(\text{SYNC}(o, p)) \\
\text{SYNCED}(o, p) \Rightarrow \text{SYNC}(o, p) \approx (o, p)
\]

In particular, if the user information in ORCID and the PTCRIS service is not consistent according to EXPORTED, running the EXPORT procedure can make them inconsistent according to IMPORTED (see Scenario 12). As such, IMPORT must be run after EXPORT to ensure that full consistency is attained.

---

\(^5\)The reason that identifier changes are allowed in PTCRIS but not in ORCID is that, while the removal/insertion of artifacts in PTCRIS are opaque to the user, the same does not apply in ORCID since it may affect the selection of the preferred work of the group.
C  Formal specification of the framework using Alloy

Alloy [2] is a lightweight formal specification language that, supported by the Alloy Analyzer, provides bounded model checking and model finding functionalities through an embedding in off-the-shelf SAT solvers. Alloy is a rich and flexible language; in this section we focus only on concepts deemed essential for the scope of this report.

An Alloy specification is developed in modules, that consist of paragraphs: signature declarations, constraints and commands. A signature declaration introduces a set of elements sharing a similar structure and properties. In Alloy such elements are uninterpreted, immutable and indivisible, and are thus denoted atoms. A signature declaration may also introduce fields, i.e. relations that connect its atoms to those of other (or the same) signatures. These are represented as sets of tuples of atoms in instances. Alloy is not restricted to binary relations, and it is not uncommon to have fields that relate three or more signatures. A signature that extends other signatures inherits their fields. It can also be contained in another signature, in which case it is simply a subset of the parent signature. Modules may also declare enumeration signatures, that contain exactly the declared atoms.

Signatures may be annotated with multiplicity keywords to restrict their cardinality, namely some (at least some elements), lone (at most one element), and one (exactly one element). The range signature in a field declaration can also be annotated with such multiplicities, to restrict the number of atoms that can be connected to each atom of the source signature. If that number is arbitrary, the special multiplicity keyword set should be used.

Facts specify properties that must hold in every instance. These may call functions and predicates, that are essentially containers for reusable expressions. Commands are used to perform particular analyses, by invoking the underlying solver. Run commands try to find instances for which the specified properties hold, while check commands try to find counter-examples that refute them. Commands can be parametrized by scopes for the declared signatures, thus bounding the search-space for the solver. If no scope is specified a default of 3 is assumed.

Consider Fig. 9 that depicts the formal specification of ORCID user profiles in Alloy. Signatures like Work, Putcode and UID declare the corresponding classes and introduce (binary) fields to represent the classes’ attributes and associations, like the putcode and uids of a work. Alloy does not have a primitive boolean type, so boolean attributes are usually represented by subset signatures containing the elements that have the attribute set to true. This is the case of the preferred attribute of work, here represented by the preferred set, a sub-set of works. The set of available external sources is defined as the enumeration type Source. This specification is then constrained by additional facts ORCID1, ORCID2 and ORCID3. Run commands would instructs the analyzer to search for instances conforming to this specification, within a specific scope for each of the signatures, for instance

```
run {} for 3 but 1 ORCID
```

Formulas in Alloy are defined in relational logic, an extension of first-order logic with relational and closure operators. Everything in Alloy is a relation, i.e. a set of tuples of atoms (with uniform arity). Signatures are unary relations (sets) containing the respective atoms and scalar values (including quantified variables) are just singleton sets. This uniformity of concepts leads to a very simple semantics. The relational logic operators also favor a navigational style of specification that is appealing to software engineers, as it resembles object-oriented languages.

The key operator in Alloy is the dot join composition that allows the navigation through fields (and relational expressions in general). For example, if \( w \) is a Work, \( w.\text{putcode} \) denotes its putcode (a scalar) and \( w.\text{uids} \) its UIDs (a set containing at least one UID). Besides composition, relational expressions can also be built using the union (+), intersection (\&), difference (-), and cartesian product (\( -> \)) operators. In particular, singleton tuples can be defined by taking the cartesian product of two (or more) scalars. Relations can also have their domain restricted to a given set (<:) and likewise for the range (\( :-> \)). For example, \( o.\text{preferred} \leftarrow o.\text{uids} \) is the binary relation that associates preferred works in an ORCID profile \( o \) with the respective putcodes. Binary relational expressions can also be reversed (\( \sim \)), extended with the transitive closure (\( ^* \)), or with the reflexive transitive closure (\( ^+ \)). For
sig Putcode, UID, Metadata {}
enum Source {PTCRIS, User, Scopus}
sig Work {
    putcode : one Putcode,
    uids : set UID,
    source : one Source,
    metadata : one Metadata
}
sig ORCID {
    works : set Work,
    preferred : set works
}
fun similar[o : ORCID, w : Work] : set Work {
    w.{w1,w2 : o.works | some w1.uids & w2.uids}
}
fact ORCID1 {
    all o : ORCID | no disj w1,w2 : o.works |
    w1.putcode = w2.putcode
}
fact ORCID2 {
    all o : ORCID | all disj w1,w2 : o.works |
    w1.source = w2.source implies no (w1.uids & w2.uids)
}
fact ORCID3 {
    all o : ORCID | all w : o.works |
    one o.similar[w] & o.preferred
}

Figure 9: Formal specification of the ORCID data model.

example, in the definition of the similar function, the reflexive transitive closure operator is used to
group all works sharing UIDs. Relational expressions may also be created by set comprehension, as
is the similar function. Finally, there are some primitive relations pre-defined in Alloy: univ denotes
the universe, i.e. the set of all tuples, none denotes the empty set, and iden the binary identity relation
over the universe.

Alloy has limited support for integers: the pre-defined Int signature contains all available integers.
In commands, the scope of Int determines the available number of bits to represent them (in two’s
complement notation). Integers can be added and subtracted with the functions plus and minus,
respectively. The default semantics for integer operations is wrap around: for example, if the scope
for Int is 3, plus[3,1] is -4. Every relation expression can have its cardinality determined with the
# operator.

Atomic formulas are built from relational expressions using inclusion (in), equality (=), or cardinal-
ity checks (besides lone, some, and one, keyword no can also be used to check if a relational
expression is empty). Formulas can be combined with conjunction (and), disjunction (or), implication
(implies), possibly associated with an else formula, equivalence (iff), and negation (not). Besides
the universal (all) and existential (some) quantifiers, Alloy also supports lone (property holds for at
most one atom), one (property holds for exactly one atom), and no (property holds for no atom) quan-
tifiers. For instance, in the ORCID3 predicate, the formula quantifies over all ORCID atoms and every
Work contained in them.
sig Key, UID, Metadata {}  
sig Production {  
  key : one Key,  
  uids : set UID,  
  metadata : one Metadata  
}  
sig Notification {  
  key : one Key,  
  uids : set UID  
}  
sig Creation extends Notification {  
  metadata : one Metadata  
}  
sig Modification extends Notification {}  
sig PTCRIS {  
  productions : set Production,  
  exported : set productions,  
  notifications : set Notification  
}  
fact PTCRIS1 {  
  all p : PTCRIS | no disj p1,p2 : p.productions |  
  p1.key = p2.key  
}  
fact PTCRIS2 {  
  all p : PTCRIS | all s : p.exported | some s.uids  
}  
fact PTCRIS3 {  
  all p : PTCRIS | all disj p1,p2 : p.productions |  
  some p1.uids & p2.uids implies p1 not in p.exported or p2 not in p.exported  
}  
fact PTCRIS4 {  
  all p : PTCRIS | no disj n1,n2 : p.notifications & Creation |  
  n1.key = n2.key  
}  
fact PTCRIS5 {  
  all p : PTCRIS | all n : p.notifications & Creation |  
  n.key not in p.productions.key  
}  
fact PTCRIS6 {  
  all p : PTCRIS | all n : p.notifications & Modification |  
  n.key in p.productions.key  
}  
fact PTCRIS7 {  
  all p : PTCRIS | all n : p.notifications & Modification |  
  some n.uids  
}  

Figure 10: Formal specification of the PTCRIS data model.
pred IMPORTED [o:ORCID, p:PTCRIS] {
  // IMPORTED1
  all w : o.works |
  some pl : p.productions+p.notifications | o.similar[w].uids in pl.uids
  // IMPORTED2
  all n : p.notifications |
  some w : o.works | o.similar[w].uids = n.uids
  // IMPORTED3
  all n : p.notifications & Creation |
  some w : o.works | o.similar[w].uids = n.uids and n.metadata = extract[o,o.similar[w]]
  // IMPORTED4
  all n : p.notifications & Creation |
  some n.uids and (all pl : p.productions+p.notifications | no n.uids & pl.uids or pl = n)
  // IMPORTED5
  all n : p.notifications & Modification |
  let mod = (p.productions)&key.(n.key) |
  some n.uids & mod.uids and n.uids not in mod.uids
  // IMPORTED6
  all w : o.works, pl : p.productions |
  some o.similar[w].uids & pl.uids and o.similar[w].uids not in pl.uids implies
  one n : p.notifications&Modification | n.key = pl.key and n.uids = o.similar[w].uids
}

Figure 11: Formal specification of the IMPORTED consistency relation.

pred EXPORTED [o:ORCID, p:PTCris] {
  // EXPORTED1
  all e : p.exported |
  one w : o.works | e.uids = w.uids and e.metadata = w.metadata and w.source = PTCRIS
  // EXPORTED2
  all w : o.works | w.source = PTCRIS implies
  one e : p.exported | e.uids = w.uids and e.metadata = w.metadata
}

Figure 12: Formal specification of the EXPORTED consistency relation.