A DATA STRUCTURE FOR THE IMPLEMENTATION OF A REFERENT TRACKING SYSTEM

Lauren M. Wishnie, MSc[1], Alexander D. Diehl, PhD[1]

[1]Department of Biomedical Informatics, University at Buffalo, Buffalo, NY, USA

Introduction

Longitudinal multimodal studies are highly useful in Alzheimer's Disease research. One of the best-known examples is the **Alzheimer's Disease Neuroimaging Initiative (ADNI)** study, which has generated a diverse body of data about people with subjective memory concern, mild cognitive impairment, and Alzheimer's Disease since 2004. Because the measurements are taken multiple times over several years, patterns in the individual biomarkers & their different combinations can be studied in greater context.

Referent tracking (RT) is a framework for the automated creation of referent unique identifiers **(RUIs)** for each non-repeatable entity (individual) referred to in the data, so that they can be explicitly tracked through time [1]. Benefits observed from RT implementations include reduced errors—everything has a unique RUI, which allows users to avoid **mistaking one entity for another**—and the integration of data from disparate sources. The changes are stored in tuples, which in the context of RT refers to ordered lists of data elements about changes that were made.

As part of an ongoing project aimed at standardizing and integrating the ADNI data, we applied a **Referent Tracking System (RTS)** to the **ADNI Knowledge Base (ADNI-KB)** so that the entities referred to within the data are trackable.

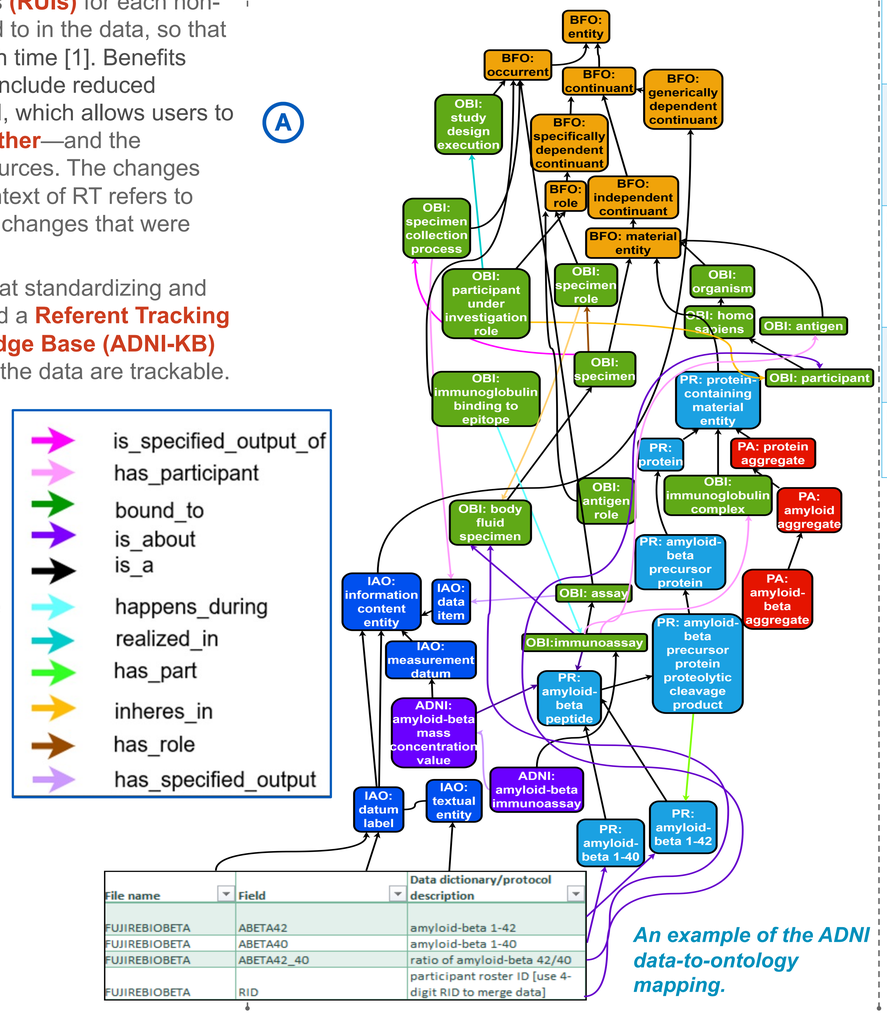
Methods

The **ADNI Ontology** is based on the ADNI data, and is axiomatized in CLIF (with an OWL version). The next phase involves linking the ADNI Ontology to the ADNI data to generate the ADNI Knowledge Base **(ADNI-KB)**. An algorithm to create the data structure for RT was written in Python (Figure D). The algorithm stores data frames of RT tuples. There are seven tuple types used in RT (Table B). The core of the algorithm is a set of **tuple generating functions (TGFs).** Each function generates tuples of a given type, and accepts a list of inputs which can come from a system administrator be generated automatically as the result of an update to the ADNI-KB.

Results

The data structure algorithm is intended to facilitate the construction of an RT database that stores records of all changes that are made to the ADNI-KB. We are currently applying the TGF method to different ADNI data types to track updates to participants' CSF/serum, imaging, medical history, and neuropsychological testing data as part of an ongoing effort to improve data quality and, by extension, patient outcomes. This tracking of entities also allows us to identify biomarker patterns at various levels of granularity—for example, changes in cognition may occur on completely different timescales from changes in serum amyloid-beta.

A



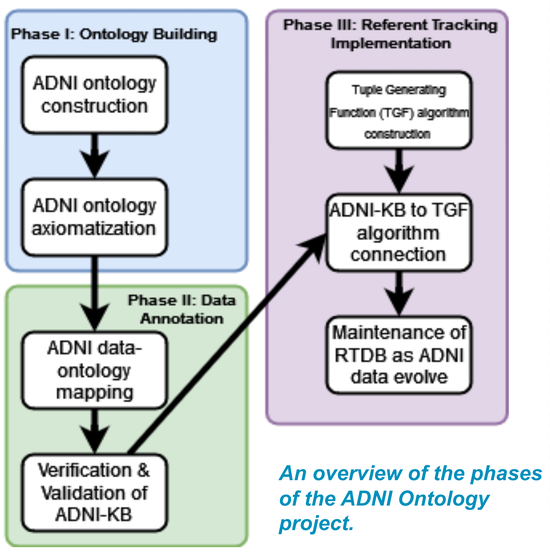
File name	Field	Data dictionary/protocol description
FUJIREBIOBETA	ABETA42	amyloid-beta 1-42
FUJIREBIOBETA	ABETA40	amyloid-beta 1-40
FUJIREBIOBETA	ABETA42_40	ratio of amyloid-beta 42/40
FUJIREBIOBETA	RID	participant roster ID [use 4-digit RID to merge data]

An example of the ADNI data-to-ontology mapping.

A *Descriptions of the tuple types in RT.*

Tuple Type	Description
D-tuple	Records that RUI was registered at a specific temporal instant (t) as an insertion (I) or as an error correction (E). If it is an insertion, a reason (R) is provided; if it is an error correction, a list of the replacement tuples (S) is provided.
F-tuple	Records the degree of confidence (C) that the author has in the faithfulness of a tuple at t.
A-tuple	Records that an RUI was assigned (A) or reserved (R) to a uniquely identifiable portion of reality (PoR) at t as a singularly unique (+SU) or non-singularly unique (-SU) identifier.
NtoN-tuple	Records whether or not it is the case (+/-) that the t at which a relation (r) exists between the non-repeatable PoRs (P) relates in manner *rT* to some t. Examples of rT values are *at, since, before*—all relations to some temporal region.
NtoR-tuple	Records whether or not it is the case (+/-) that the t at which an instantiation relation (r) exists between a repeatable (RUI_r) and non-repeatable (RUI_n) PoR relates in manner *rT* to some t. Examples of rT values are *at, since, before*—all relations to some temporal region.
$NtoR^-$ -tuple	Records that it is the case that the t at which the relation does not exist between the non-repeatable PoR (RUI_p) and any of the non-repeatable PoRs in which a repeatable PoR (RUI_r) is repeated relates in manner *rT* to temporal region t_r.
NtoC-tuple	Records whether or not it is the case (+/-) that the t at which the concept code from the concept system (RUI_{cs}) is an accurate code for the non-repeatable PoR (RUI_p).
NtoI-tuple	Records whether or not it is the case that the t at which RUI c considers a particular descriptor type (nt_j) for PoR (RUI_p) relates in manner *rT* to temporal region t_r.

B



An overview of the phases of the ADNI Ontology project.

Additionally, in the future, we will develop methods for updates to the ADNI data uploaded to the ADNI website to be efficiently added to the ADNI-KB so that the RT system can track the changes in real time.

C

```
def NI_tuple_generator(NI_tuple_values):
    NI_rui = "NtoI_" + str(uuid.uuid1())
    NI_timestamp = '{:%Y-%m-%d %H:%M:%S}'.format(datetime.datetime.now())
    NI_tuple_values.insert(0, NI_rui)
    NI_tuple_values.insert(5, NI_timestamp)
    NI_lol.append(NI_tuple_values)
    NI_df = pd.DataFrame(NI_lol, columns=['ni_rui', 'case_status', 'descriptor type',
                                          'descriptor', 'rT', 'temporal region'])
    return NI_df

def A_tuple_generator(A_tuple_values):
    A_rui = "A_" + str(uuid.uuid1())
    A_timestamp = '{:%Y-%m-%d %H:%M:%S}'.format(datetime.datetime.now())
    A_tuple_values.insert(0, A_rui)
    A_tuple_values.insert(4, A_timestamp)
    A_lol.append(A_tuple_values)
    A_df = pd.DataFrame(A_lol, columns=['rui_a', 'rui_por','assigned/reserved',
                                        'SU', 'temporal_instant'])
    return A_df

def NC_tuple_generator(NC_tuple_values):
    NC_rui = "NtoC_" + str(uuid.uuid1())
    NC_timestamp = '{:%Y-%m-%d %H:%M:%S}'.format(datetime.datetime.now())
    NC_tuple_values.insert(7, NC_rui)
    NC_tuple_values.insert(6, NC_timestamp)
    NC_lol.append(NC_tuple_values)
    NC_df = pd.DataFrame(NC_lol, columns=['nc_rui','case_status', 'relation', 'rui_concept_system',
                                          'rui_nonrep', 'concept_code', 'rT', 'temporal_region'])
    return NC_df
```

Three Tuple Generating Functions (TGFs) that create the tuples. The tuples are stored in data frames according to type.

The functions generate and store the tuples, first by generating an RUI for each particular (including each tuple). The RUI for the ontology term is not a randomly generated ID, but rather the term label with the source ontology as a prefix.

D

	rui_a	rui_por	assigned/reserved	SU	temporal_instant
0	A_b6f019c3-3e26-11ed-938d-144f8a97c60a	b6f019c2-3e26-11ed-8887-144f8a97c60a	A	+SU	2022-09-27 01:39:11
1	A_b6f324ec-3e26-11ed-b65a-144f8a97c60a	6506	A	+SU	2022-09-27 01:39:11
2	A_b6f55cd8-3e26-11ed-a237-144f8a97c60a	7052	R	+SU	2022-09-27 01:39:11
3	A_b6f7b588-3e26-11ed-84bc-144f8a97c60a	6290	R	+SU	2022-09-27 01:39:11

	nr_rui	case_status	rui_nonrep	inst_relation	rui_rep	rT	temporal_region
0	NtoR_432b8fae-3e29-11ed-86b0-144f8a97c60a	+	6056	BFO:instanceOf	OBI: participant	at	2022-09-27 01:57:25
1	NtoR_432db254-3e29-11ed-811d-144f8a97c60a	+	7052	BFO:instanceOf	OBI: specimen	at	2022-09-27 01:57:26
2	NtoR_4330e334-3e29-11ed-a2b0-144f8a97c60a	+	6290	BFO:instanceOf	OBI: immunoassay	at	2022-09-27 01:57:26

Examples of dataframes constructed by the data structure algorithm, all of which together form the RT database (RTDB).

Conclusion

We hope to make the ADNI data model generalizable to non-ADNI knowledge bases that serve a similar purpose to ADNI. We continue to work towards automating the RTDB updating process, as well as verification and validation of our system.

References

1. Ceusters, W. The Place of Referent Tracking in Biomedical Informatics. Unpublished work. 2021.

Acknowledgements

Many thanks to the Department of Biomedical Informatics, Dr. Alexander Diehl for his guidance, and Dr. Werner Ceusters for his advice & all his work developing Referent Tracking.

University at Buffalo The State University of New York

Department of Biomedical Informatics
Jacobs School of Medicine and Biomedical Sciences
buffalo.edu