ACE TUTORIAL (Version 1.0)

Goals
The goal for this tutorial is to familiarize users with basic queries and data extraction using ACE. By the end of this tutorial you should be comfortable with ACE query structure. Complete language specifications can be found at http://shahlab.stanford.edu/ACE.

What is ACE?
The Advanced Cohort Engine (ACE) is a web tool that uses a temporal query language to search and extract patient data. Data sources include Stanford EHRs and external claims databases. Stanford EHR data covers 2009-2017 and includes records on over 2 million patients.

ACE uses a data model that structures patient records in major categories including demographics, diagnosis codes, measurements, procedures, and clinical note annotations.

How do I access ACE?
ACE is available at https://ace.stanford.edu. You will need to be connected to the VPN to use this site. To access this site you will need to follow the steps for onboarding affiliates (https://shahlab.stanford.edu/onboarding_affiliates) and Stanford VPN Access (https://uit.stanford.edu/service/vpn).

ACE Temporal Query Language
The ACE temporal query language allows users to search patient data by restricting values for any record type, for example to find all patients with a diagnosis of liver cancer or all patients who were prescribed the drug metformin. The query language supports boolean and temporal commands to combine queries for specific patient features into more complex queries, for example to find all female patients of Asian descent who have had a diagnosis of hypertension. ACE boolean commands are `AND`, `OR` and `NOT`. ACE temporal commands include `INTERSECT` (find patients with specified features occurring at the same time), `UNION` (find patients with specified features, any of which may have occurred), and `SEQUENCE` (find patients with specific features that occur in sequence, one after the other). Additional commands allow for searching by medical constructs such as `HISTORY OF`, `NEVER HAD`, or `NO HISTORY OF`.

In ACE records, each patient's record is organized on a timeline. The timeline is built of features (eg labs, vitals, hospital visits) and their associated dates. The beginning of the timeline is the first time the patient appears in the EHR and the end is his/her last appearance in the EHR. Dates are measured as the number of days since the patient's birth. On the timeline, all data features are associated with a start date/time and an end date/time. The duration of time between the start and end is the *time interval (TI).* If the start date/time is the same as the end date/time, that is considered a *time point (TP)* ***(Figure 1)***. For example, consider an inpatient hospital admission for a patient. If the inpatient admission occurs from January 1, 2010, to

January 4, 2010, then the data feature `VISIT TYPE = ‘Inpatient’` would have a TI = 4 days. If during that admission, the patient received a diagnosis of diabetes (`ICD9=250`), then the data feature `ICD9=250` would also have a TI = 4 days. If a patient had a WBC level drawn on February 1, 2011, then the data feature `LAB= ‘WBC’` would have TP = February 1, 2011. If two records of the same event type occur in overlapping time intervals, for example a record of an inpatient visit with a start date of January 12, 2010, and an end date of January 14, 2010, and a second inpatient visit with a start date of January 13 2010 and an end date of January 16, 2010, ACE will create a *computed time interval (CTI)* for an inpatient visit that spans January 12 to January 16, 2010, and this single CTI will be returned by a query for `VISIT TYPE = ‘Inpatient’`. When a command returns a CTI you will no longer have access to the original start or end time.

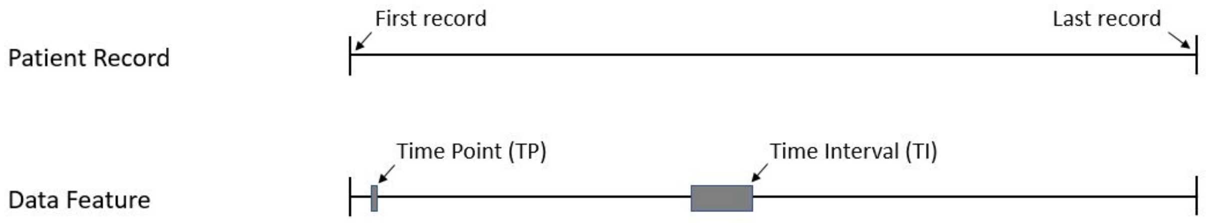

Figure 1. Example of a patient record, with a time point (TP) and time interval (TI) for a generic data feature.

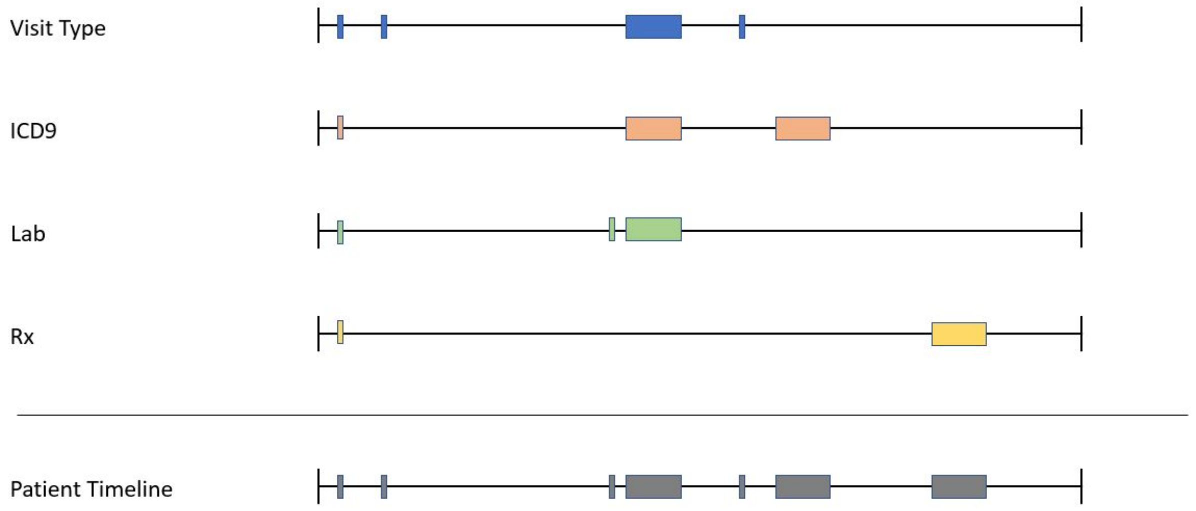

Figure 2. Example of a patient timeline showing separate data features (e.g. Visit Type, ICD9). TIs and TPs from individual data features are combined to make up the patient timeline.

Note that each data feature type (e.g. visit type, ICD9, CPT, labs, vitals) can be associated with both TIs and TPs. In some situations, this may not make clinical sense (e.g. `VISIT TYPE =`

`“Outpatient”` and TI = 9 days). This is a quirk of the data and the product of aggregating multiple data sources.

The commands you use in a search determine the result type that is returned. Results can either be a set of booleans (true/false), which are converted to entire patient timelines, or time intervals (a range of time offsets for which a given feature is true). If a temporal command is used in a search, then the search will return patient time intervals. If only boolean commands are used in a search, boolean values are converted to entire patient timelines and returned. When temporal data is exported from ACE, it is ordered by TI or TP. This means that each row is a data feature associated with a TI or TP.

Commands

1) To start, let’s find all the women in the database. Type the command below into the lower-left text box, then click the ‘Query’ button.

```
GENDER="FEMALE"
```

After running the query, notice that the top bar shows how many patients met the query criteria (approximately 1.2 million in this case). Basic summary statistics, including predominant gender, age, number of encounters and length of encounters are shown at the right. Scrolling down shows common ICD9 codes, CPT codes, drugs and labs for the retrieved cohort as compared to all patients in the database. Clicking the ‘Download Dataset’ button will download a list of the patient IDs (PIDs) of the retrieved cohort.

2) Start a new query and find all the Native Americans in the database.

```
RACE="NATIVE AMERICAN"
```

3) Start a new query and find all the patients aged 40-45 years.

```
AGE(40 years, 45 years)
```

The input is `AGE(X,Y)`, with the default unit of age being *minutes* e.g. `AGE(4, 10)` returns all the patients 4-10 minutes old. Other units of age must be specified e.g. `AGE(4 months, 10 months)` or `AGE(4 years, 10 years).` `AGE` does not return all patients who ever had a record indicating a given age, but rather the TI when they were in the given age.

4) Find all the diabetics, as defined by ICD9=250.

```
ICD9=250
```

You will notice when you start typing `ICD9=` ACE will provide you with a suggestion box of possible values for your query. These entries are the 10 most frequent possible value you could provide ordered from most to least frequent. You could also type “Diabetes” and see the same suggestions. This feature is in place for many commands.

5) Find all the Native American women, aged 40-45 years, with diabetes. Query terms can be combined using `AND` or `INTERSECT`.

```
AND(GENDER="FEMALE", RACE="NATIVE AMERICAN", AGE(40 years, 45 years),
ICD9=250)
```

`AND` is a Boolean command that returns all instances of Native American women, aged 40-45, who have had a diagnosis of diabetes (`ICD9=250`) at *any point*. The output of `AND` (and all other Boolean commands) is the entire timeline of patients who meet the search criteria. **(Figure 3)**

```
INTERSECT(GENDER="FEMALE", RACE="NATIVE AMERICAN", AGE(40 years, 45 years),
ICD9=250)
```

`INTERSECT` is a temporal command that returns all instances of Native American women, aged 40-45, who have carried a diagnosis of diabetes (`ICD9=250`) *between the ages of 40-45*. `INTERSECT` only returns instances where all the characteristics are met *at the same time*. The output of the `INTERSECT` command are TIs, not the entire patient timeline.

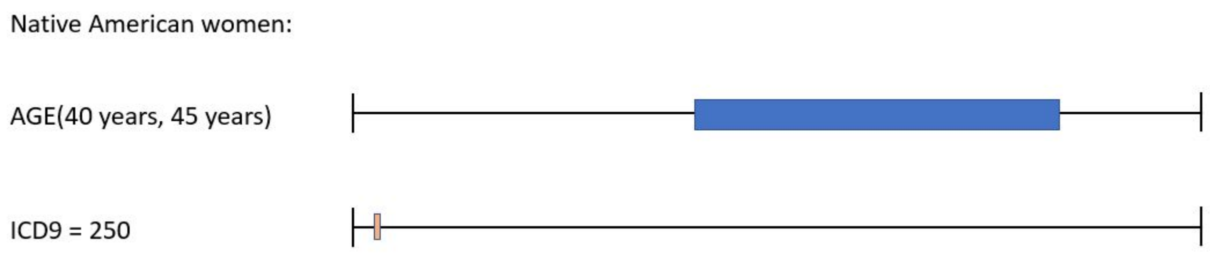

Figure 3. Shows a Native American woman who had a diagnosis of diabetes (ICD9 = 250) before her 40th birthday and no diagnosis of diabetes during ages 40-45. Using the AND command would include this woman; using the INTERSECT command would not.

6) Find all Native American women with diabetes as identified by ICD9 or ICD10 codes, who have received metformin or insulin.

```
INTERSECT(GENDER="FEMALE", RACE="NATIVE AMERICAN", OR(ICD10=E08-E13,
ICD9=250), OR(RX=6809, Rx=5856))
```

The `OR` command is a Boolean command that looks for any of the conditions within the parentheses to be met at any time on the patient's timeline. Note the `ICD10` command that accepts ranges of input codes. The `RX` command accepts RxNorm identification numbers for medications. Both RX and ICD9/ICD10 commands can map typed words to appropriate codes.

You can also search prescribed drugs by class using the ATC code hierarchy, which will return any records where a drug that is a member of the specified class was prescribed. For example to find records of any lipid modifying drug prescription, the query is:

```
ATC="C10A"
```

7) Now let's find the first time patients were diagnosed with either diabetes (`ICD9=250`) or abnormal glucose tolerance test (`ICD9=648.8`):

```
FIRST MENTION(UNION(ICD9=250, ICD9=648.8))
```

This query introduces two new commands, `FIRST MENTION` and `UNION`. Both are examples of commands that return a *computed time interval (CTI),* instead of a TI or TP. This means that the time interval is computed from the input TIs and thus may be different from the input TIs (Figure 4).

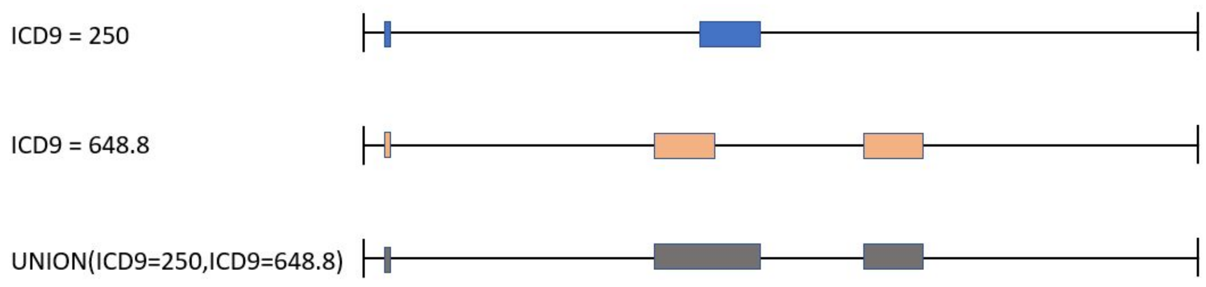

Figure 4. Illustration of UNION command between ICD9=250 and ICD9=6488. Resultant CTIs in gray.

`UNION` combines multiple TIs. This is different from the `OR` command in that `UNION` outputs CTIs, whereas `OR` outputs the entire patient timeline. Notice above that the output CTIs (gray) are different from the two input TIs (blue and orange). Note that when a command returns a CTI you no longer have access to the original start or end time.

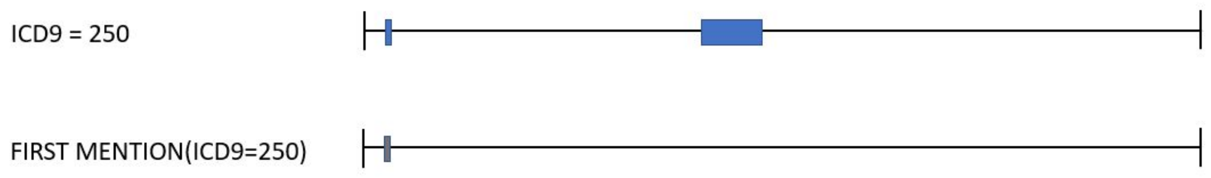

Figure 5. Illustration of FIRST MENTION command. Resultant CTI in gray.

`FIRST MENTION` outputs the TI associated with first appearance of the indicated code (`ICD9 = 250` in the above example). Note that the output TI (gray) is different than the input TIs (blue), showing that `FIRST MENTION` returns a CTI. `LAST MENTION` behaves similarly.

Note that `INTERSECT` (seen above in Example 5) also returns a CTI.

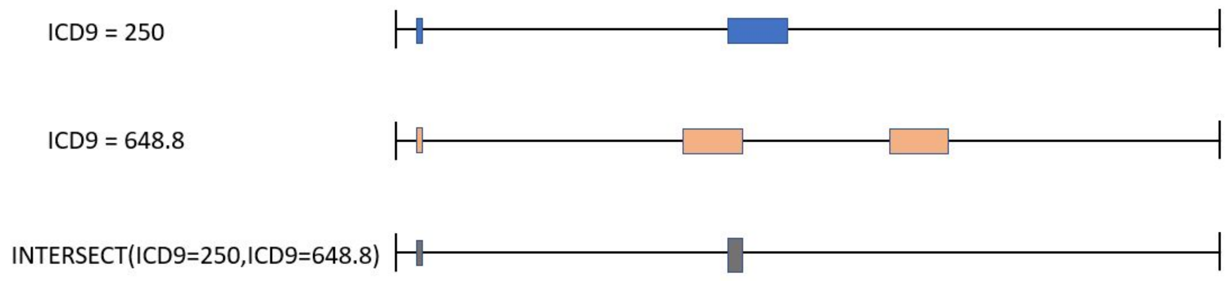

Figure 6. Illustration of INTERSECT command between ICD9=250 and ICD9=648.8. Resultant CTIs in gray.

`INTERSECT` outputs a CTI only where indicated codes (in the above example, `ICD9 = 250` and `ICD9 = 648.8`) overlap.

8) Find any patient who underwent spinal fusion or had a note that mentioned ‘Scoliosis’ prior to the age of 21.

```
INTERSECT(AGE(1, 20 years), UNION(CPT=22612, CPT=22558, TEXT="scoliosis"))
```

The `CPT` (Current Procedural Terminology) command can be used to find procedures. Like ICD9/ICD10 and RX, it can map typed words to appropriate CPT codes. `TEXT` command searches available notes and explicitly excludes negation (i.e. excludes ‘Not concerning for scoliosis.’) and family history (i.e. excludes ‘Patient has a family history of scoliosis.’). The `!TEXT` command finds negated references, and the `~TEXT` command finds family history references.

9) Find all progress notes that mention ‘diabetes’ and ‘poor compliance.’

```
NOTE(NOTE TYPE="progress notes", AND(TEXT="diabetes", TEXT="poor
compliance"))
```

The `NOTE` command allows you to search the text within a note type, specified by `NOTE TYPE`.

10) Find all of the patients who received Ceftriaxone (`RX = 2193`) within the 5 days preceding a laparoscopic appendectomy (`CPT=44970`).

```
SEQUENCE(RX=2193*, CPT=44970)+(-1, -5 days)
```

`SEQUENCE` is a sequential command that returns TIs based on their temporal relationship to other TIs. It's a tricky command that can take multiple parameters (see full language specifications), but its basic form is `SEQUENCE(X*,Y)`. This says return any X that precedes a Y. `SEQUENCE(X,Y*)` returns any Y that is preceded by an X.

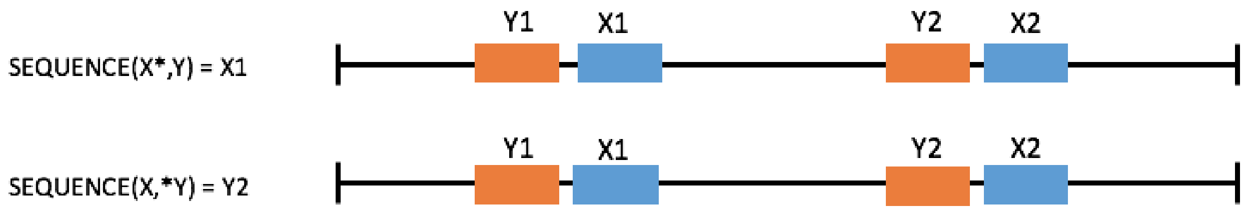

Figure 7. Illustration of SEQUENCE command. Note that the input with the asterisk determines output.

`SEQUENCE(X*,Y)+(-A,-B)` will return any X that precedes Y as long as X falls within the range defined by (-A, -B). `SEQUENCE(X*,Y)-(-A,-B)` will return any X that precedes Y as long as X does not fall within the range defined by (-A, -B).

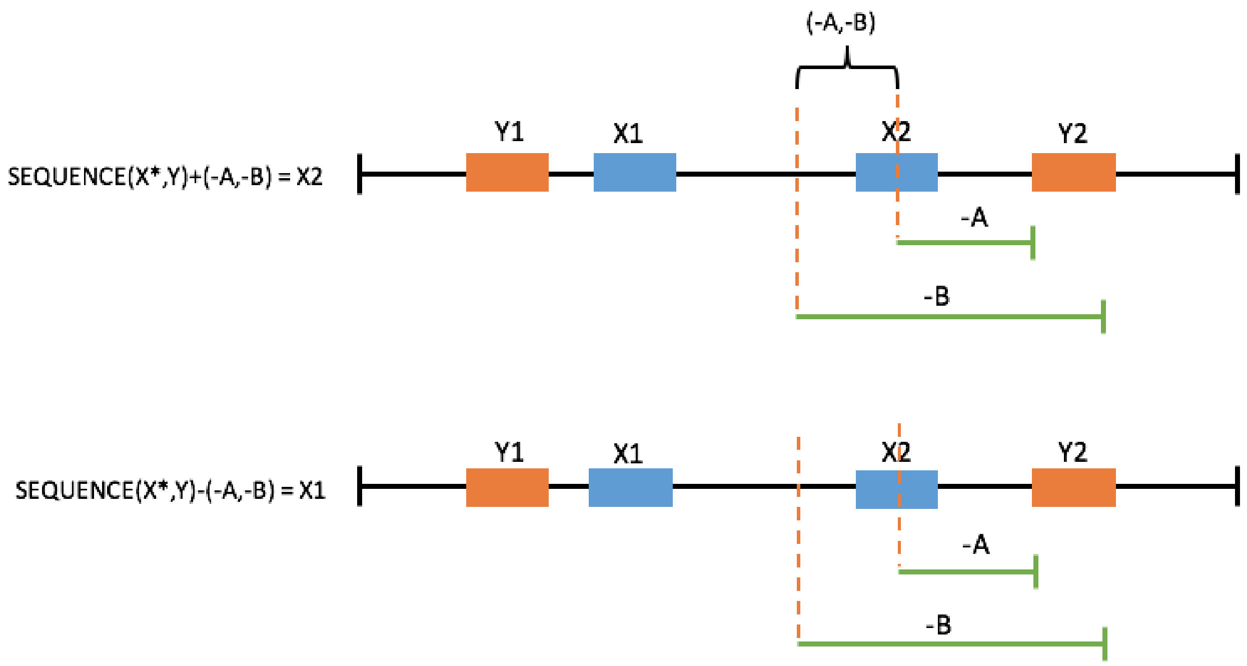

Figure 8. Illustration of SEQUENCE command. Note that in the top example, X2 is before Y2 and falls within the range defined by (-A,-B) and is thus returned. In the bottom example, X1 is before Y2 and falls outside the range defined by (-A,-B) and is thus returned.

11) Find patients who had a history of DVT of the lower extremity (`ICD9=453.4`) prior to having total knee arthroplasty (`CPT = 27447`).

```
INTERSECT(HISTORY OF(ICD9=453.4), CPT=27447))
```

`HISTORY OF (X)` returns at TI from the first mention of X until the end of the record. `NO HISTORY OF(X)` returns at TI from the beginning of patient's chart to the first instance of X or the patient's entire timeline if no X occurred.

12) Find patients that are diabetic (`ICD9=250`), but have never been diagnosed with abnormal glucose tolerance test (`ICD9=790.2`).

```
VAR example_var_1 = ICD9=250
VAR example_var_2 = ICD9=790.2
VAR example_var_3 = DIFF($example_var_1, $example_var_2)
$example_var_3
```

For more complicated queries, variables can be defined, referenced and run. The variable is created with the `VAR` command. The variable is then referenced using '`$`,' which can be used to place the variable in a separate function or to run a query with that variable.

Note the use of the `DIFF(X,Y)` command. This is a cohort level command that outputs a list of patient IDs that are present in X, but not in Y. The order of X and Y is important -- `DIFF(Y, X)` will return the patient IDs that are present in Y, but not in X. `SAME(X,Y)` is a similar cohort level command that outputs a list of patient IDs that are present in both X and Y. The order of X and Y is not important for `SAME`.

Another temporal command that is very useful is the `DURATION` command, which is used to specify that a time interval must be of a specific length. For example, to find inpatient visits of at least 5 days, the command is:

```
DURATION(VISIT TYPE="inpatient", SINGLE, 5 DAYS, MAX)
```

Here, '`SINGLE`' denotes that each single inpatient visit must be at least 5 days. Putting '`ALL`' here instead would mean that in total, all inpatient visits for a given patient must sum to at least 5 days.

13) Obtain the entire patient timeline since the first instance of receiving a diagnosis of multiple sclerosis.

```
INTERVAL(ICD9=340, END(TIMELINE))
```

The interval command constructs a TI using two parameters. The first is the first parameter is the start of the TI and the second is the end of the TI. Interval parameters can be numeric, in which case it constructs an interval from the time points specified.

<u>**Output Operations**</u>

After running a query in ACE, typically users want to output the data for further analysis. The most common output command is the `CSV` command. The `CSV` command outputs a CSV (comma-separated values) file that has a defined set of columns. The values displayed in the rows are based on the input values and can be changed by the user.

```
VAR example_var_1= INTERSECT(ICD9=250, ICD9=V16)
CSV($example_var_1, ICD9=V16)
```

After running the above query, press the 'Download Dataset' button. This will download a CSV file where each row is a TI associated with `ICD9=V16`. Other information contained in the file include patient ID (PID), Year, Start Age In Days (of TI), and End Age In Days (of TI). If a specific ICD9 value was not provided (e.g. `CSV($example_var_1, ICD9)`), the file would contain all TIs associated with all ICD9 values for the identified patients. Data features that can be output in CSV commands are: `ICD9, ICD10, LABS, VITALS, CPT, RX`.

The timeframe of variables output by the CSV command can be specified by using the `TIME` modifier. `TIME` accepts any commands to define the output TI of interest. The bounds of the TI specified in `TIME` can be manipulated by the `EXTEND BY` command.

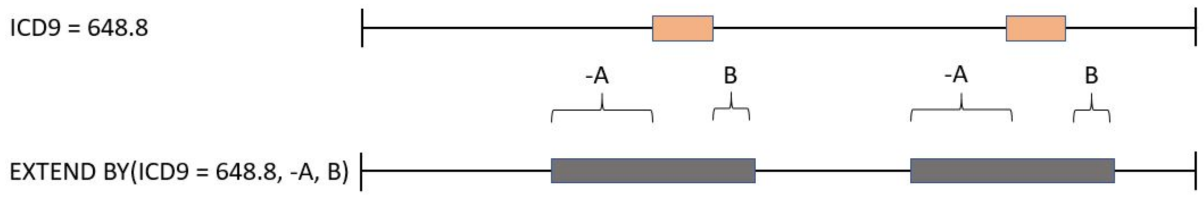

Figure 9. Illustration of EXTEND BY command. Note that the bounds of the TIs are adjusted based on the inputs A, B.

`EXTEND BY(TI, start_time, end_time)` takes an input TI and adjusts its start by the amount of time set by start_time (negative is toward the past, positive is toward the future) and adjusts its end by the amount of time set by end_time.

```
VAR example_var_1 = INTERSECT(ICD9=250, ICD9=V16)
CSV($example_var_1, TIME=EXTEND BY($example_var_1, -10 days, 0), ICD9=250,
ICD9=V16)
```

`OUTPUT(X)` is another output command. `OUTPUT(X)` produces only PIDs and the TI associated with that PID. Output columns are PID, Start Age in Days, End Age in Days.

```
VAR example_var_1= INTERSECT(ICD9=250, ICD9=V16)
OUTPUT($example_var_1)
```

Common Pitfalls

1) ACE follows an order of operations.

```
UNION(FIRST MENTION(ICD9=443), FIRST MENTION(ICD9=250))
```

is not the same as

```
FIRST MENTION(UNION(ICD9=443, ICD9=250)
```

Double check the order of nested logic commands to make sure the functionality is what you expect.

2) ACE interprets the underlying data and can remove patients based on what is queried. When adding features to a query patients might be dropped from analysis from a simpler query.

```
//335 patients
INTERSECT(YEAR(2007,2007), UNION(ICD9=572.2, ICD10=K72))

// 336 patients
INTERSECT(YEAR(2007,2007), UNION(ICD9=572.2))
```

3) `INVERT` does not equal `NOT`

The `NOT` command is Boolean and will return the patients who do not meet the criteria. `NOT(ICD9=250)` = all patients who did not have a mention of diabetes in their records. `INVERT` is a temporal command that will return the time interval where the condition is not true (using the patient's timeline as a reference).

This brings up an interesting point. `INVERT(ICD9=250)` will give you the entire timeline for any patient who did not have `ICD9=250` in their timeline. For the patients who did have an `ICD9=250` they will return only the time intervals where the `ICD9=250` was not true.

4) Combining temporal commands with Boolean parameters should be avoided.

```
BEFORE(AND(ICD9=250.50, RX=161), ICD9=410*)
```

In the above command the `AND` Boolean operator will return the entire timeline of patients meeting the criteria. The `BEFORE` command will try to then find instances where the entire timeline came before and `ICD9=410`; this returns a value but *not* what is desired. ACE will give you a warning when you try to do this.

5) `INTERSECT` requires that two or more events happened at the exact same time. For many events in the EMR this might be too strict. It is often necessary to use the `EXTEND BY` function to broaden the time interval.

6) Often you care about whether the first point in a patient's timeline a condition is true. The `FIRST MENTION` command will allow you to pull out this information.

7) The `AGE` command defaults to minutes. Always provide a unit of time when using it.

```
AGE (18,24) = AGE(18 MINUTES, 24 MINUTES)
```

8) In general phenotyping of disease does not necessarily mean that the disease is present. For example, presence of `ICD9=250` does not mean person is diabetic. If you can add a more stringent method to identify patients that will lead to stronger results. For example, diabetic patients have records of glycemic control, or Hemoglobin A1C measurement.

<u>**Practice Questions**</u>

1) Find all women diagnosed with amyloidosis via ICD9 or ICD10 codes. Output a file that contains all the procedures (ie CPT codes) those women have received.

2) Find all women diagnosed with amyloidosis via ICD9 or ICD10 codes. Find all of those women who received a heart transplant *after* their diagnosis of amyloidosis. Output a file that contains the age of women when she received the heart transplant.

3) Find all women diagnosed with amyloidosis via ICD9 or ICD10 codes. Of these, find all the women who received morphine in the 30 days preceding their *first diagnosis* of amyloidosis in the *outpatient* setting. Output a file that contains the first diagnosis of amyloidosis, the prescription of morphine within the preceding 30 days and the outpatient visit where the diagnosis was made.

4) Find all patients whose timelines are at least one year in length. Use a combination of `INTERVAL`, `RECORD START`, `RECORD END`, and `DURATION` for this.

5) Example consult question 1:
Clinical question: In patients at least 18 years old, and prescribed ibuprofen, is there any difference in peak blood glucose after treatment compared to patients prescribed acetaminophen?

We define "acetaminophen" as the first mention of `RX=161`. We define "ibuprofen" as the first mention of `RX=5640`. For either drug, we restrict to patients prescribed only that drug and never prescribed the other. We define "peak blood glucose" as the highest blood glucose measurement within 7 days of treatment. We also measure "mean pre-treatment blood glucose" as the mean blood glucose from the 30 days prior to treatment, restricted to patients who had at least 3 measurements pre-treatment.

6) Example consult question 2:

Clinical question: In female patients at least 18 years old and diagnosed with a urinary tract infection, and prescribed a sulfonamide antibiotic, is there any difference in peak blood glucose after treatment compared to patients prescribed a cephalosporin antibiotic or penicillin derivative antibiotic?

We define "urinary tract infection" as the first occurrence of either ICD9 599 or ICD10 N39.0. We define "cephalosporin antibiotic" as any of ATC code J01DB, J01DC, or J01DE. We define "sulfonamide antibiotic" as any of `RX=10180`, `RX=10829`, `RX=10171`, `RX=9524`, or `RX=10207`. We define "penicillin derivative antibiotic" as ATC code "J01C". For a given antibiotic, we restrict to patients prescribed only that antibiotic and never either of the other two types. We restrict our analysis to the first mention of urinary tract infection, and the first recorded prescription that intersects with that time interval. We define "peak blood glucose" as the highest blood glucose measurement within 7 days of diagnosis and treatment. We also measure "mean pre-treatment blood glucose" as the mean blood glucose from the 30 days prior to treatment, restricted to patients who had at least 3 measurements pre-treatment.

Solutions to Practice Questions

1)

```
VAR F_Amyloidosis = INTERSECT(GENDER="FEMALE", UNION(ICD9=277.3,
ICD10=E85))
CSV($F_Amyloidosis, CPT)

For this particular question either UNIOR/OR would have identical results.
```

2)

```
VAR F_HO_Amyloidosis=INTERSECT(GENDER="FEMALE", HISTORY
OF(UNION(ICD9=277.3,ICD10=E85)))

CSV(INTERSECT($F_HO_Amyloidosis,CPT=33945), CPT=33945)
```

3)

```
VAR var1 = INTERSECT(GENDER="FEMALE", UNION(VISIT TYPE="outpatient", VISIT
TYPE="appointment", VISIT TYPE="office visit"), FIRST
MENTION(UNION(ICD9=277.3, ICD10=E85)))
```

```
VAR var2 = BEFORE(RX=7052, $var1*) + (0, -30 days)

CSV($var2, TIME=EXTEND BY($var2, -30 days, 0), VISIT TYPE="outpatient",
VISIT TYPE="appointment", VISIT TYPE="office visit", ICD9=277.3, ICD10=E85,
RX=7052)
```

4)

```
VAR timeline = INTERVAL(RECORD START, RECORD END)
VAR timeline_min_1yr = DURATION($timeline, SINGLE, 1 YEAR, MAX)
```

5)

```
VAR age = AGE(18 years, MAX)

VAR acetaminophen = FIRST MENTION(RX=161)

VAR ibuprofen = FIRST MENTION(RX=5640)

VAR blood_glucose = LABS("2345-7")

VAR acetnoibu = SEQUENCE($age, INTERSECT($acetaminophen, NOT($ibuprofen))*)

VAR ibunoacet = SEQUENCE($age, INTERSECT($ibuprofen, NOT($acetaminophen))*)
```

6)

```
VAR age = AGE(18 years, MAX)

VAR female = GENDER="FEMALE"

VAR uti = FIRST MENTION(UNION(ICD9=599, ICD10=N39.0))

VAR cohort = INTERSECT($age, $female, $uti)

VAR sulfa = UNION(RX=10180, RX=10829, RX=10171, RX=9524, RX=10207)

VAR penicillin = ATC="J01C"
VAR ceph = UNION(ATC="J01DB", ATC="J01DC", ATC="J01DC", ATC="J01DE")

VAR sulfa_only = INTERSECT($sulfa, NOT($penicillin), NOT($ceph))

VAR penicillin_only = INTERSECT($penicillin, NOT($sulfa), NOT($ceph))

VAR ceph_only = INTERSECT($ceph, NOT($penicillin), NOT($sulfa))

VAR sulfa_cohort = FIRST MENTION(INTERSECT($cohort, $sulfa_only))

VAR penicillin_cohort = FIRST MENTION(INTERSECT($cohort, $penicillin_only))

VAR ceph_cohort = FIRST MENTION(INTERSECT($cohort, $ceph_only))
```

Frequently asked questions:

1. What does it mean when the query results indicate skipped patients?

There are few reasons why certain patients are not eligible for evaluation and are skipped. The main reason is that the query contains a feature for which the patient contains data that are not correct (data points before the date of birth, data points in the future, etc.). For these data point the entire feature is marked as invalid and if the query attempts to evaluate the patient with this feature, patient is skipped from evaluation. (Example: ICD9=250.50 contains a data point in the future. The code 250.50 is marked as invalid and when attempted to evaluate, the whole patient is skipped. Other ICD9 codes can still be evaluated for the patient).

Vitals and Labs containing invalid time events are simply removed, since their temporal persistence is low, unlike ICD9, CPT, RX codes.

The second reason a patient is skipped is when trying to evaluate a NOT or INVERT query. If the patient does not contain any events of that type, it does not mean that the specific event did not occur, it is just an example of missing data and should not be evaluated.

(Example: Patient's records do not contain any medications. Query: NOT(RX=161) will skip the patient, since it is unclear why there are no medication records. If the patient would contain at least 1 medication record, query would succeed).