LazyTAP: In Praise of Laziness

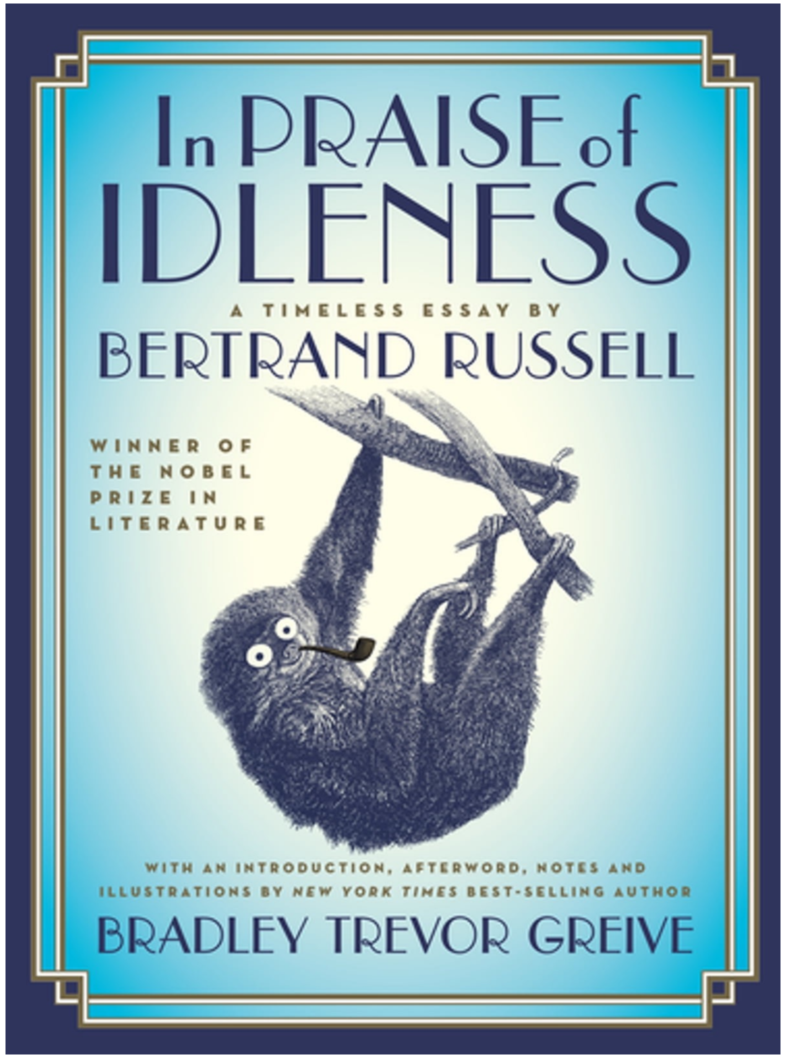
Mohammad M. Ahmadpanah Daniel Hedin Andrei Sabelfeld

CHALMERS UNIVERSITY OF TECHNOLOGY

Mälardalens universitet

February 17, 2023

KTH Royal Institute of Technology


In PRAISE of IDLENESS
A TIMELESS ESSAY BY
BERTRAND RUSSELL
WINNER OF THE NOBEL PRIZE IN LITERATURE
WITH AN INTRODUCTION, AFTERWORD, NOTES AND ILLUSTRATIONS BY NEW YORK TIMES BEST-SELLING AUTHOR
BRADLEY TREVOR GREIVE

- "... the road to happiness and prosperity lies in an organized diminution of work."
- "We should increase leisure and produce only what makes for better lives."

Trigger-Action Platform (TAP)

- Connecting otherwise unconnected services and devices
- Trigger event comes, the app performs an action
- Popular TAPs: IFTTT, Zapier, Power Automate

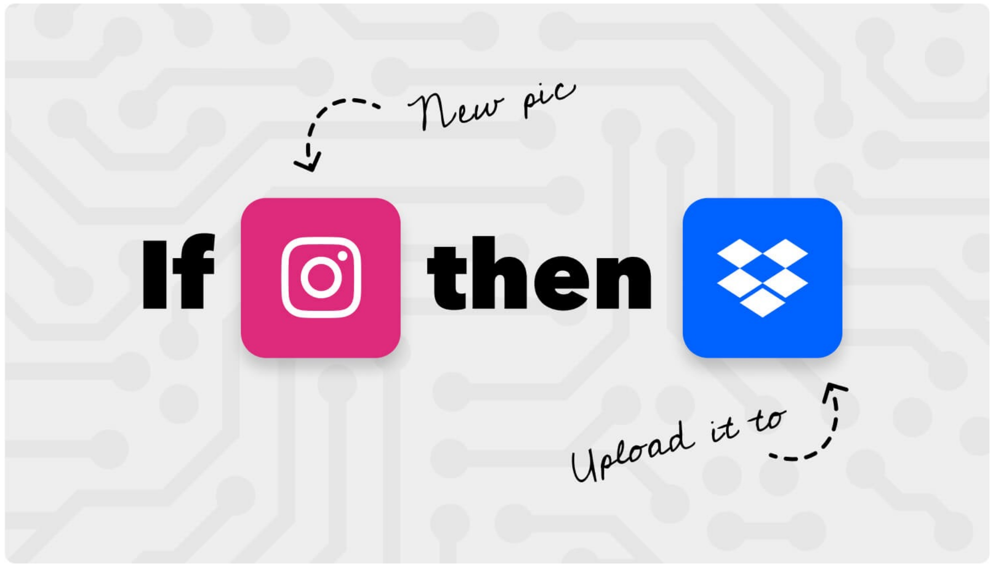


Image: © IFTTT

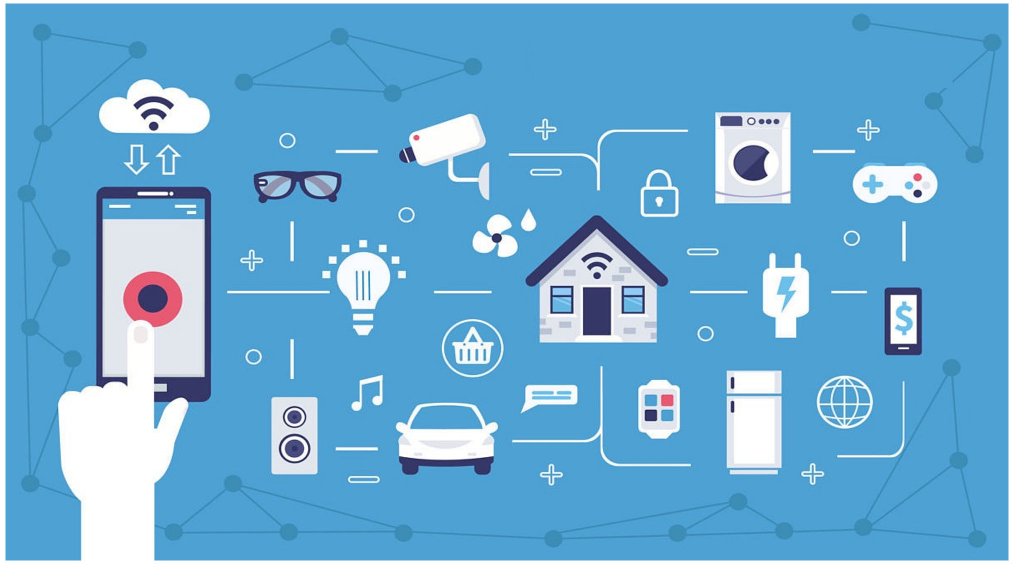


Image: © Irina Strelnikova / Adobe Stock

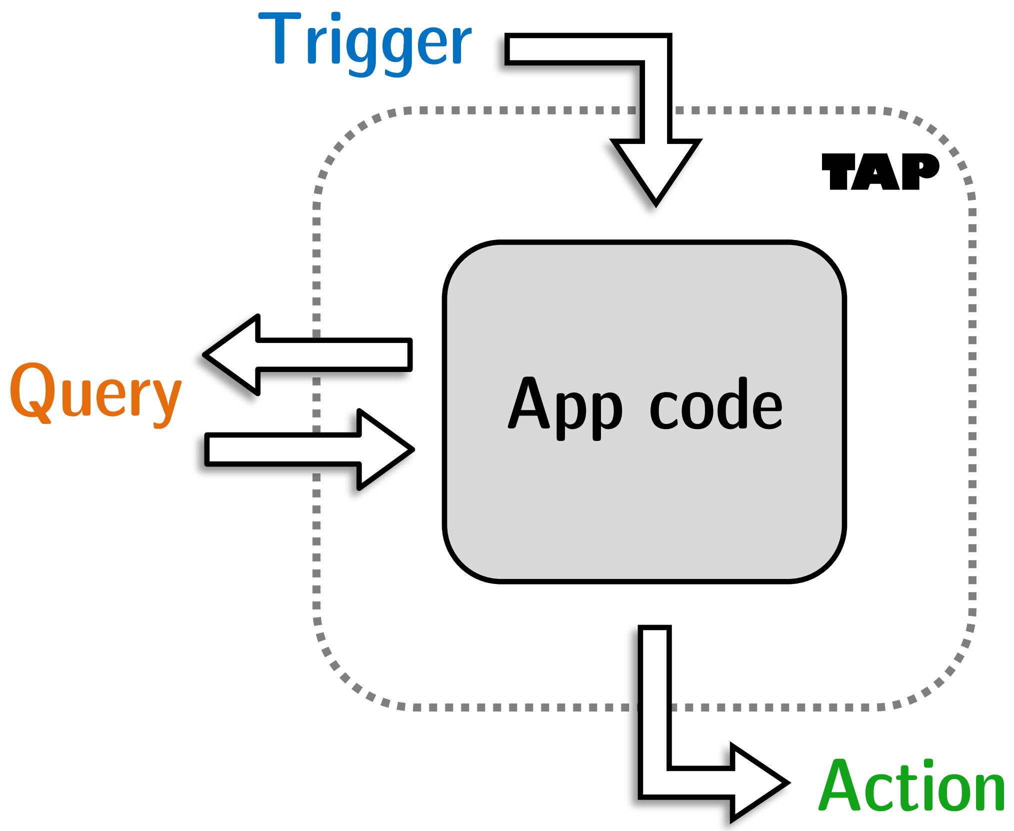
IFTTT example: Amir’s app

Trigger

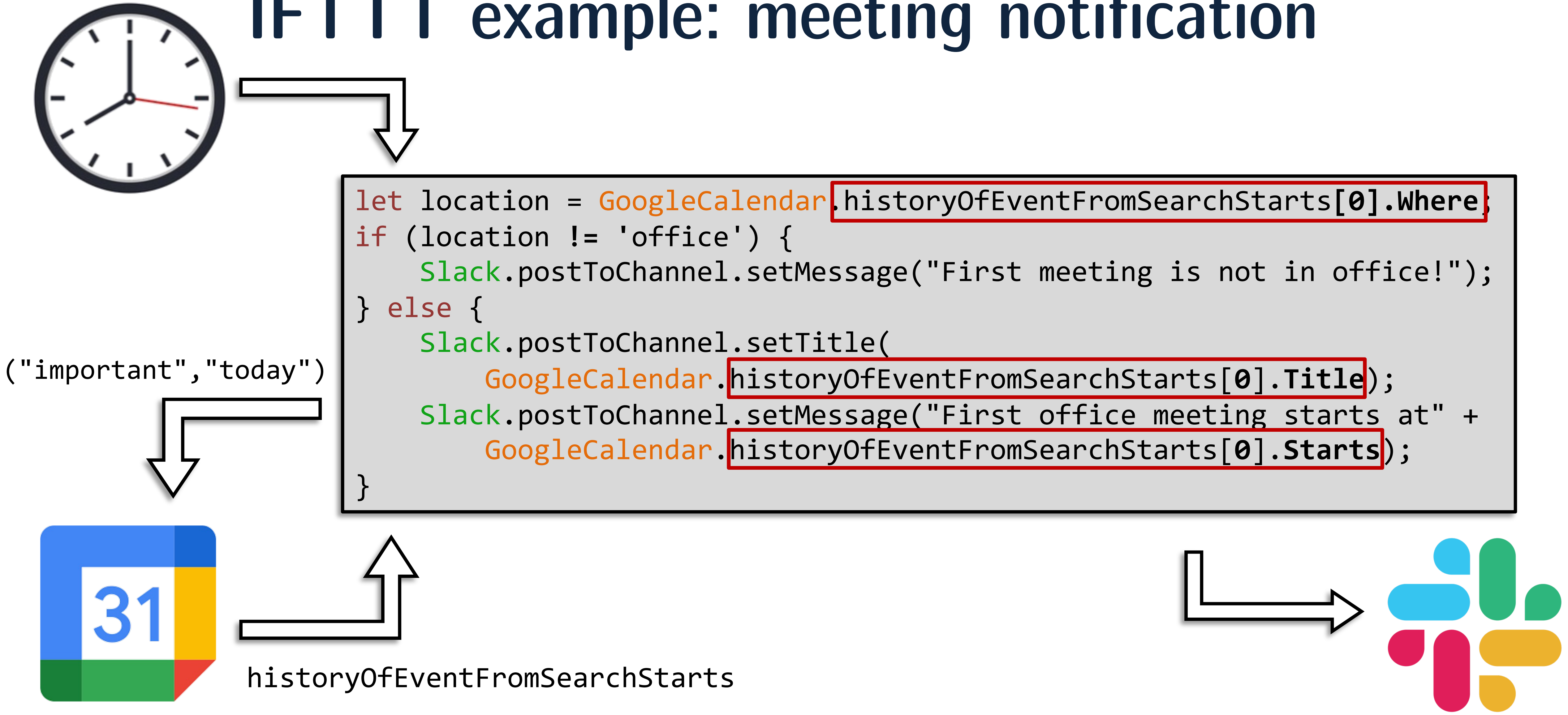
```
// app filtercode
if (Email.sendIftttAnEmail.From == "Musard") {
    Slack.postToChannel.setMessage(
        Email.sendIftttAnEmail.Subject +
        Email.sendIftttAnEmail.Body +
        Email.sendIftttAnEmail.AttachmentUrl);
} else {
    Slack.postToChannel.skip();
}
```

Action

IFTTT

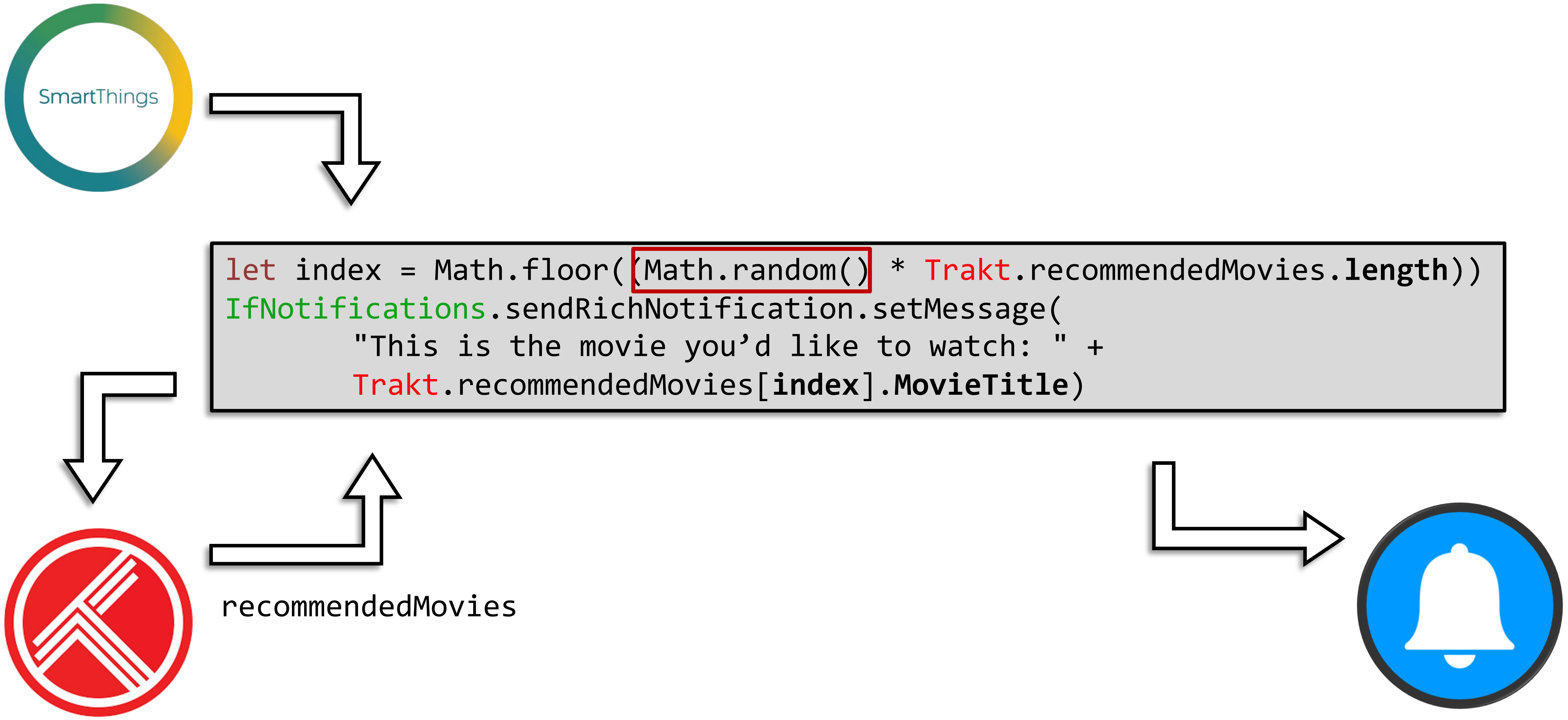


IFTTT example: meeting notification



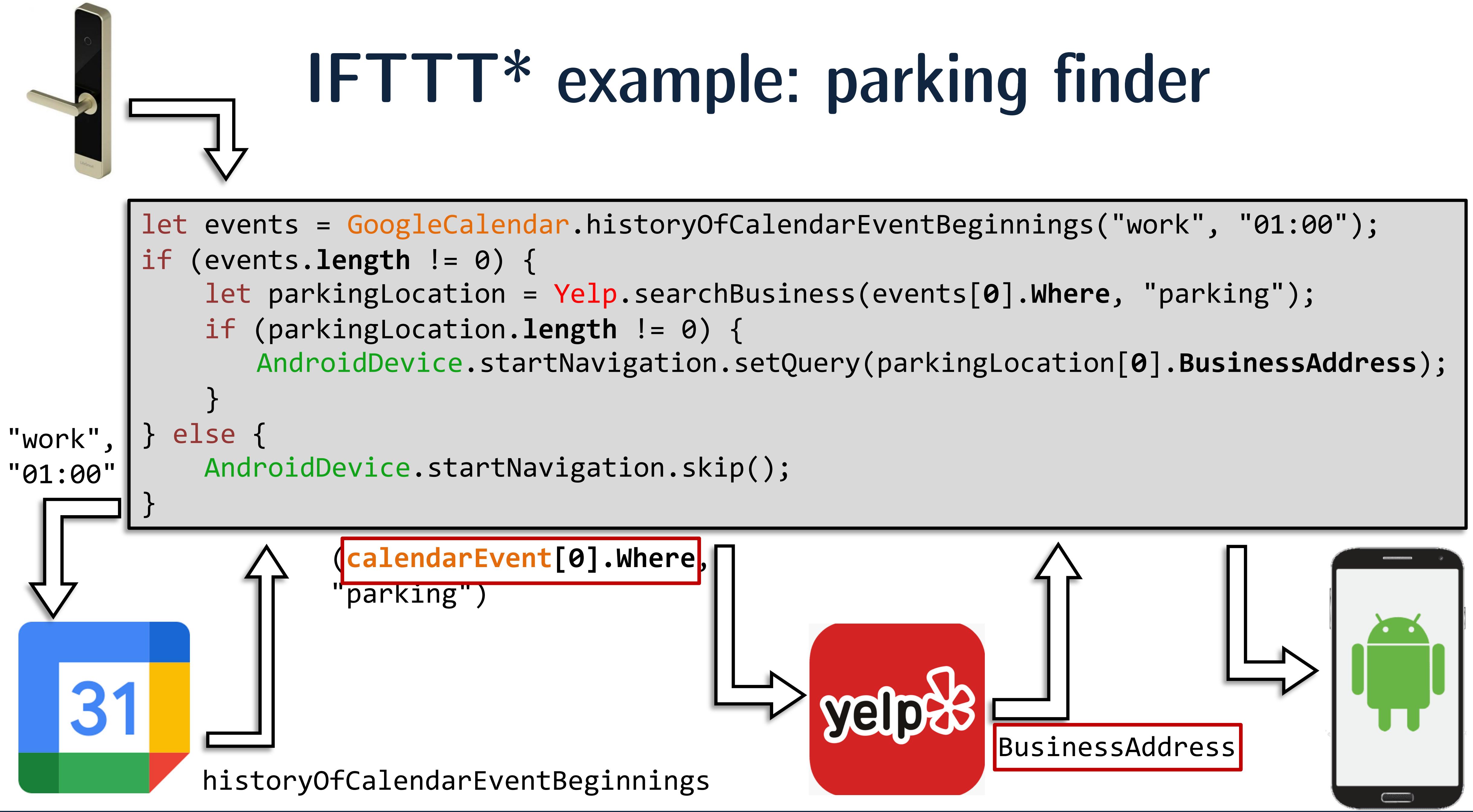
```
let location = GoogleCalendar.historyOfEventFromSearchStarts[0].Where;
if (location != 'office') {
    Slack.postToChannel.setMessage("First meeting is not in office!");
} else {
    Slack.postToChannel.setTitle(
        GoogleCalendar.historyOfEventFromSearchStarts[0].Title);
    Slack.postToChannel.setMessage("First office meeting starts at" +
        GoogleCalendar.historyOfEventFromSearchStarts[0].Starts);
}
```

IFTTT example: movie recommender



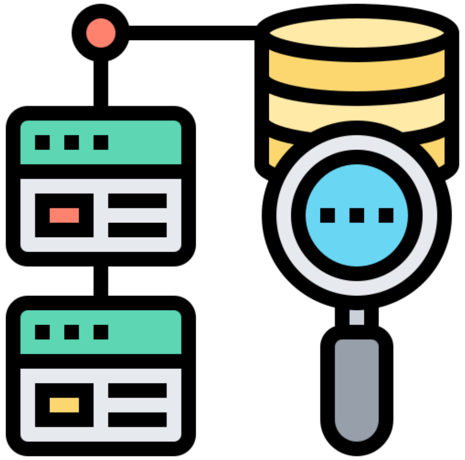
```
let index = Math.floor(Math.random() * Trakt.recommendedMovies.length))
IfNotifications.sendRichNotification.setMessage(
      "This is the movie you’d like to watch: " +
      Trakt.recommendedMovies[index].MovieTitle)
```

IFTTT* example: parking finder



```
let events = GoogleCalendar.historyOfCalendarEventBeginnings("work", "01:00");
if (events.length != 0) {
    let parkingLocation = Yelp.searchBusiness(events[0].Where, "parking");
    if (parkingLocation.length != 0) {
        AndroidDevice.startNavigation.setQuery(parkingLocation[0].BusinessAddress);
    }
} else {
    AndroidDevice.startNavigation.skip();
}
```

Queries



- Additional data *source* of apps
- In IFTTT:
  - Can be *multiple* for each app
  - Can depend on *trigger* data
    - Query chain is not possible
  - Fetch *all* the queries first, then execute the app code
- A general TAP:
  - Queries might depend on any data source (trigger/query)
    - Supporting query chains

Data minimization

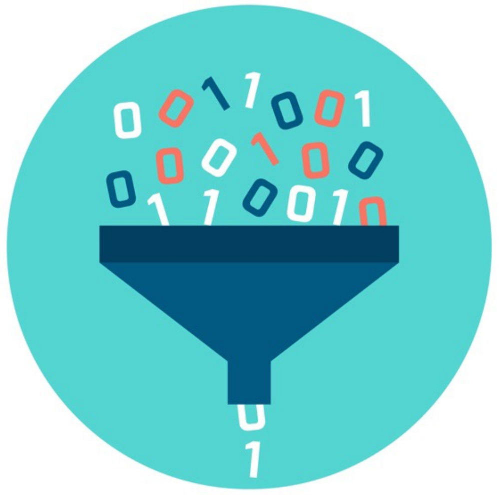
- Only necessary data should be collected for the specific purpose the user consented
- IFTTT's approach: Attribute-level overprivilege
  - Fetching *all* the attributes from trigger/query services *no matter* what the app's execution needs!
  - Email attributes: {`AttachmentTemporaryUrl`, `From`, `Body`, `BodyHTML`, `Subject`, `AttachmentUrl`, `ReceivedAt`}
  - "A trigger endpoint should return (by default) the 50 most recent events, regardless of whether or not we have seen them before"
- Goal: Transmit the *minimal* set of attributes *required* for the app's execution

minTAP [USENIX'22]

- Minimization wrt: ill-intended TAP
  - May deviate from the protocols to steal *more* user data than needed
- Minimizing attributes of *trigger* data
  - *Preprocessing* approach
- Modes: Static and Dynamic
  - Static: Aggregates all the attributes <u>existing</u> in the app source code
  - Dynamic: <u>Pre-runs</u> the app code on the trigger service to identify attributes accessed in the execution
  - Trusted client outside of the TAP (dependency analysis and app integrity)

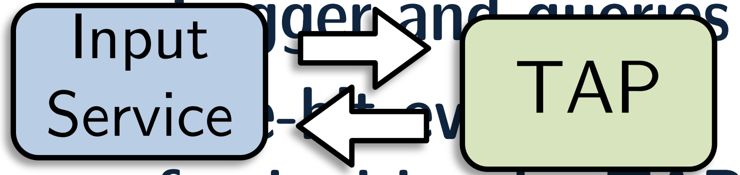
minTAP (cont.)

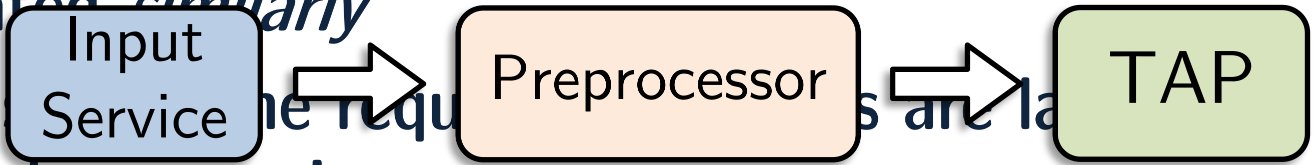
- Notions of data independence
  - Value changes of a subset of attributes has no effect on the output
  - Regular: Output never depends on what values an attribute takes
  - Run: Output depends on values of certain attributes
- Static minTAP
  - Regular independence + support for queries?
- Dynamic minTAP
  - Run independence + no trigger if action skips



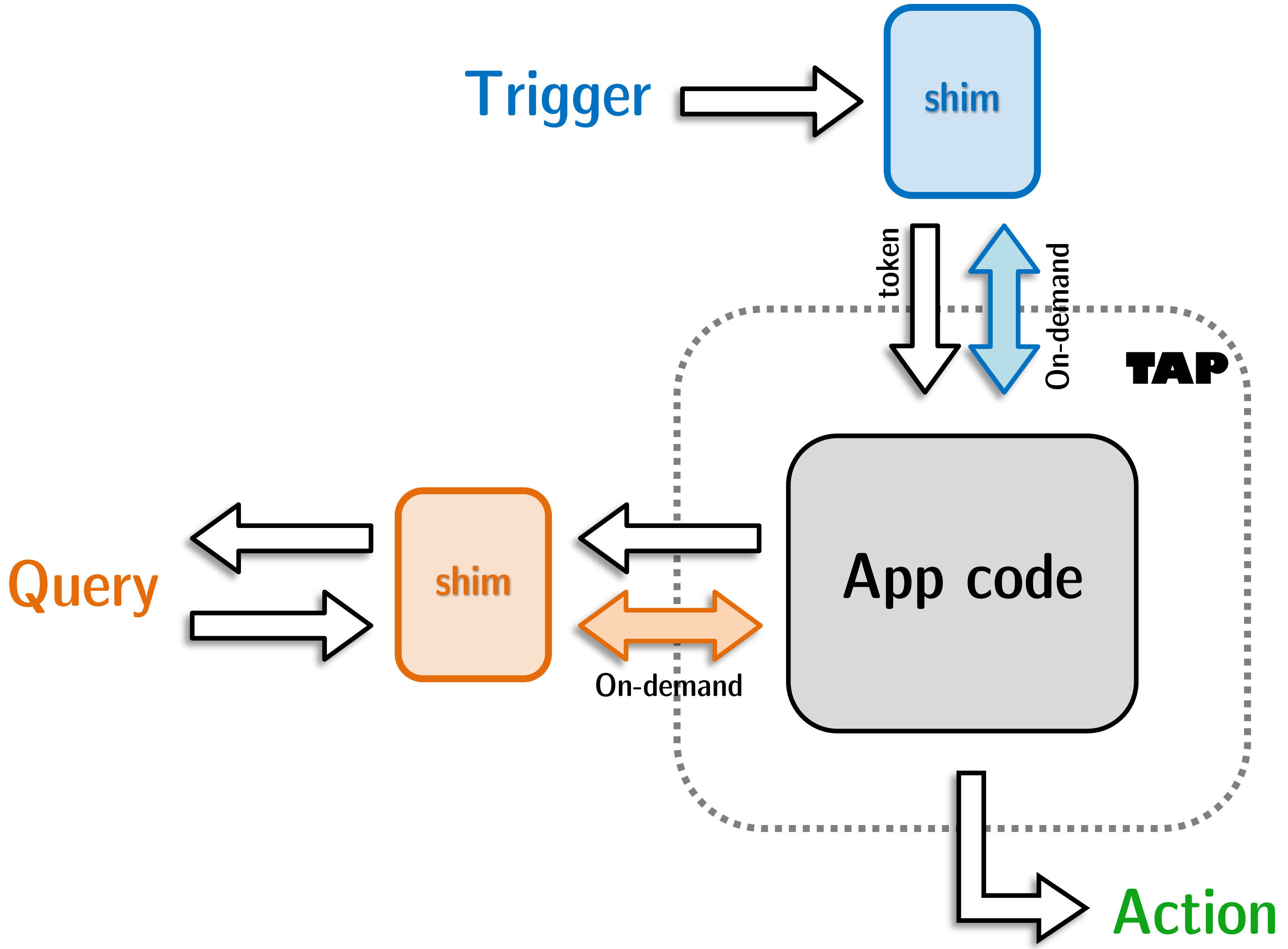
LazyTAP: Data minimization by design

- Minimization wrt: willing-to-minimize TAP
  - Incentivized to support the principle of data minimization
- Fetch-on-need
  - Attribute is fetched from service only if it is *accessed* in the execution
  - *On-demand* vs. *preprocessing* minimization
- Data source unification
  - Trigger and queries are treated *similarly*
  - e-b ev trigger i he requ s are la
    fetched by the TAP during the execution





LazyTAP

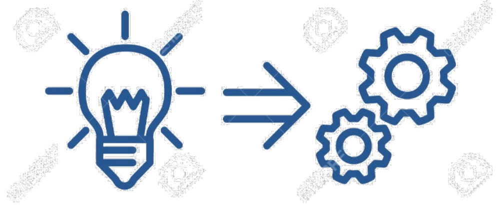

Trigger
shim
token
On-demand
TAP
Query
shim
App code
On-demand
Action

LazyTAP (cont.)

- Which parts are changing?
  - TAP: Executes app when trigger token received, then fetching data *on-the-fly*
  - Trigger/Query services: Shim layers or using trusted mediators
    - Caching mechanism
- App code remains as is
  - A lazy version of the runtime for app, using the same APIs

LazyTAP: Implementation

- Elements:
  - Remote Trigger: fetch-on-need trigger data
  - Lazy Query: deferred computation of query, providing fetch-on-need query data
  - Lazy Projection: delayed projection *via thunking*
- Deferred computation by thunking
  - Using proxies or accessor properties, queries as classes
- Queries can depend on *any* data source (trigger or query)



LazyTAP: Demo

- Examples
  - Meeting notification
  - Movie recommender
  - Parking finder

Evaluation

App Id	Distinctive pattern	Total attributes (IFTTT)	Static minTAP	LazyTAP
MeetNotif	Sensitive independent query	3 + (7 * CalendarLength)	3	1 | 3
MovieRec	Nondeterministic query, skip on time	3 + (7 * TraktLength)	TraktLength+1	1 | 2 | 3 | 4
ParkFind	Conditional query chain, skip on queries	4 + (7 * CalendarLength) + (7 * YelpLength)	4	1 | 3 | 4

Minimization: 95% over IFTTT; 38% over minTAP

LazyTAP: Formalism

- Core language: While language with objects

$$\begin{aligned} e &::= v \mid x \mid e \oplus e \mid f(e) \mid e[e] \mid \{\} \mid T \mid Q(k, e) \mid A(m) \\ &\quad \mid \boxed{() \Rightarrow e} \\ i &::= x \mid i[e] \\ c &::= \mathit{skip} \mid i := e \mid \mathit{if}\ e\ \mathit{then}\ c\ \mathit{else}\ c \mid \mathit{while}\ e\ \mathit{do}\ c \mid c; c \end{aligned}$$

- Supporting lazy computation, lazy query, and remote objects
  - Trigger and query data represented by *remote objects*, caching data on the first access
- Semantics: strict and lazy (also in Coq)

LazyTAP: Formalism (cont.)

- Extensional equivalence
  - Executing on *equivalent* memories, lazy app behaves the *same* as strict
- Minimality
  - Lazy semantics fetches *no more attributes* than what the strict semantics demands

Strict semantics

$$\Gamma = \langle t, q, a \rangle$$

$$\Gamma \models (c, E, H) \rightarrow_s (E, H)$$

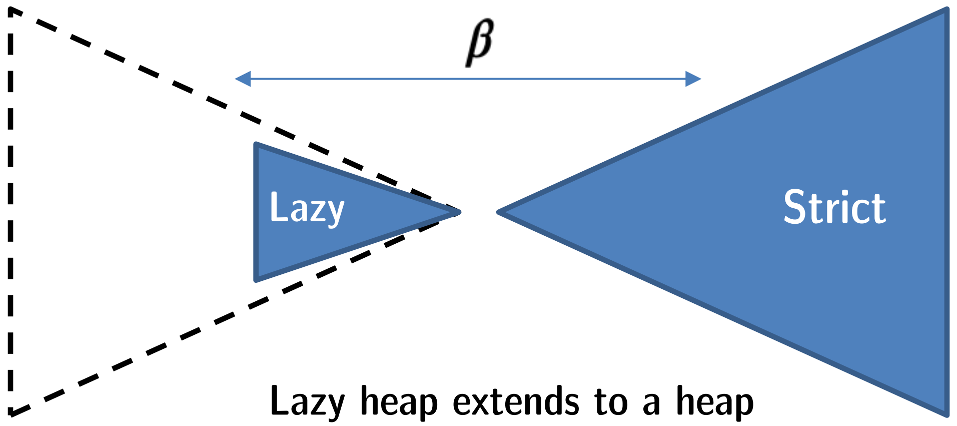
Lazy semantics

$$\Gamma = \langle (t, F_t), q, a \rangle$$

$$\Gamma \models (c, E, H) \rightarrow_l (E, H)$$

LazyTAP: Formalism (cont.)

- Extensional equivalence
  - Contexts are *isomorphic* under β
  - Mapping refs to refs and remote refs to refs bijectively
- Lazy context $\simeq_\beta$ Strict context
  - Perform all deferred computations,
  - Fetch all attributes from the remote objects
  - The resulting lazy context is isomorphic to the strict context



Lazy heap extends to a heap isomorphic to strict heap

$$((t, F_t), q, a, E, H) \simeq_\beta (t, q, a, E, H)$$

LazyTAP: Formalism (cont.)

- LazyTAP apps model IFTTT apps

$$
\begin{gathered}
\forall c, c', \beta_1, \mathit{\Gamma}, \mathit{E}_1, \mathit{R}_1, \mathit{H}_1, \Gamma, E_1, H_1 E_2, H_2. \\
\boxed{(\mathit{\Gamma}, \mathit{E}_1, \mathit{R}_1, \mathit{H}_1) \simeq_{\beta_1} (\Gamma, E_1, H_1)} \land \\
c' = compileL2S(c) \land \\
\boxed{\Gamma \models (c', E_1, H_1) \rightarrow_s (E_2, H_2)} \Rightarrow \\
\exists \beta_2, \mathit{E}_2, \mathit{R}_2, \mathit{H}_2. \boxed{\mathit{\Gamma} \models (c, \mathit{E}_1, \mathit{R}_1, \mathit{H}_1) \rightarrow_l (\mathit{E}_2, \mathit{R}_2, \mathit{H}_2)} \land \\
\beta_1 \subseteq \beta_2 \land \\
\boxed{(\mathit{\Gamma}, \mathit{E}_2, \mathit{R}_2, \mathit{H}_2) \simeq_{\beta_2} (\Gamma, E_2, H_2).}
\end{gathered}
$$

LazyTAP: Formalism (cont.)

- LazyTAP apps model <u>only</u> IFTTT apps

$$\begin{gathered}
\forall c, c', \beta_1, \varGamma, \mathsf{E}_1, \mathsf{R}_1, \mathsf{H}_1, \Gamma, E_1, H_1 \mathsf{E}_2, \mathsf{R}_2, \mathsf{H}_2. \\
\boxed{(\varGamma, \mathsf{E}_1, \mathsf{R}_1, \mathsf{H}_1) \simeq_{\beta_1} (\Gamma, E_1, H_1)} \land \\
c' = compileL2S(c) \land \\
\boxed{\varGamma \models (c, \mathsf{E}_1, \mathsf{R}_1, \mathsf{H}_1) \rightarrow_l (\mathsf{E}_2, \mathsf{R}_2, \mathsf{H}_2)} \Rightarrow \\
\exists \beta_2, E_2, H_2. \boxed{\Gamma \models (c', E_1, H_1) \rightarrow_s (E_2, H_2)} \land \\
\beta_1 \subseteq \beta_2 \land \\
\boxed{(\varGamma, \mathsf{E}_2, \mathsf{R}_2, \mathsf{H}_2) \simeq_{\beta_2} (\Gamma, E_2, H_2)}.
\end{gathered}$$

LazyTAP: Formalism (cont.)

- Precision of LazyTAP

  “LazyTAP is at least as precise as sound *preprocessing* minimization techniques”

  – Starting from the minimized initial environments,
  – Correctness of the static/dynamic minimization techniques,
  – The app execution successfully maps the minimized initial environments to a final environment,
  – Every strict environment has a lazy counterpart

LazyTAP vs. minTAP

Approach	Minimization wrt	Apps without queries	Apps with queries
IFTTT	None	Push all, no minimization guarantees	
Static minTAP	Ill-intended TAP	Regular independence (Input-unaware minimization)	
Dynamic minTAP	Ill-intended TAP	Run independence (Input-sensitive minimization) + No attribute when skip/timeout	N/A
LazyTAP	TAP willing to minimize	Run independence (Input-sensitive minimization) + Real-time behavior-preserving (incl. arrays, nondeterminism)	

LazyTAP takeaways

Be *lazy*, be *minimized!*

- Data minimization o

nt queries
- *On-demand*, *input-s* *eserving* minimization
- Changes:
  - Trigger/query servic ches
    - No code exeuction o
  - TAP: supporting fet ntime for the app