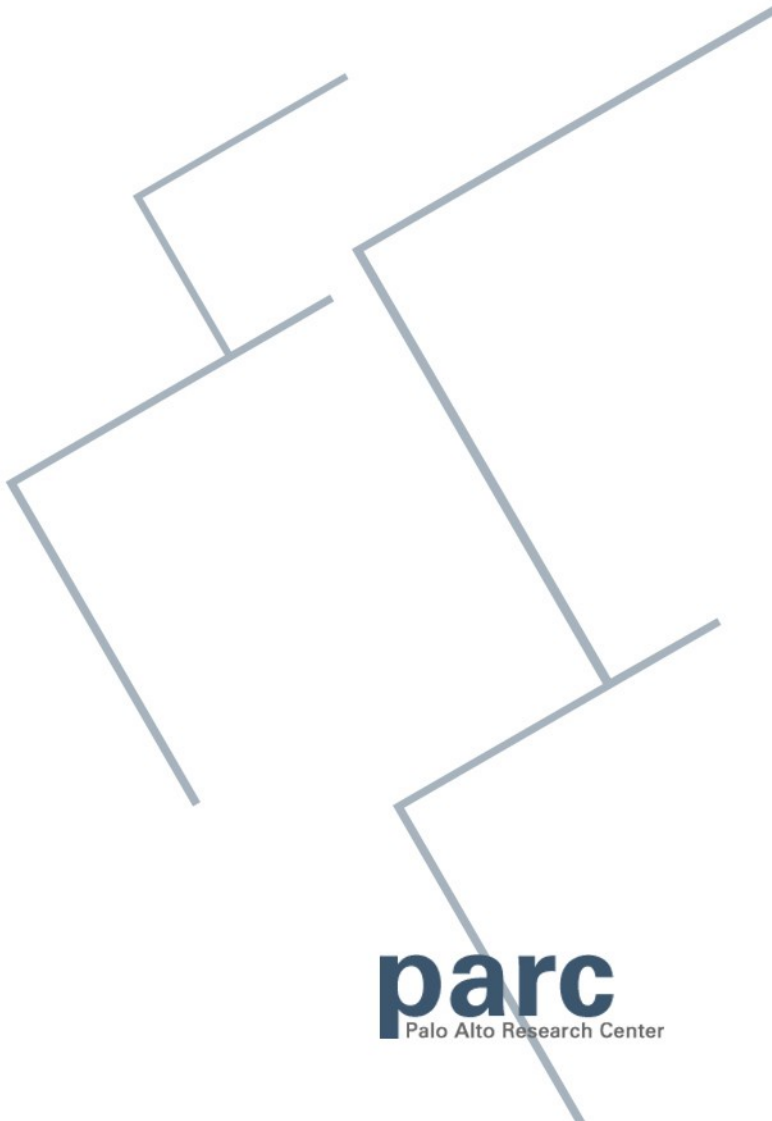
Constructive Access Control : Revisited?

Valeria de Paiva
Intelligent Systems Lab
PARC



parc
Palo Alto Research Center

Outline

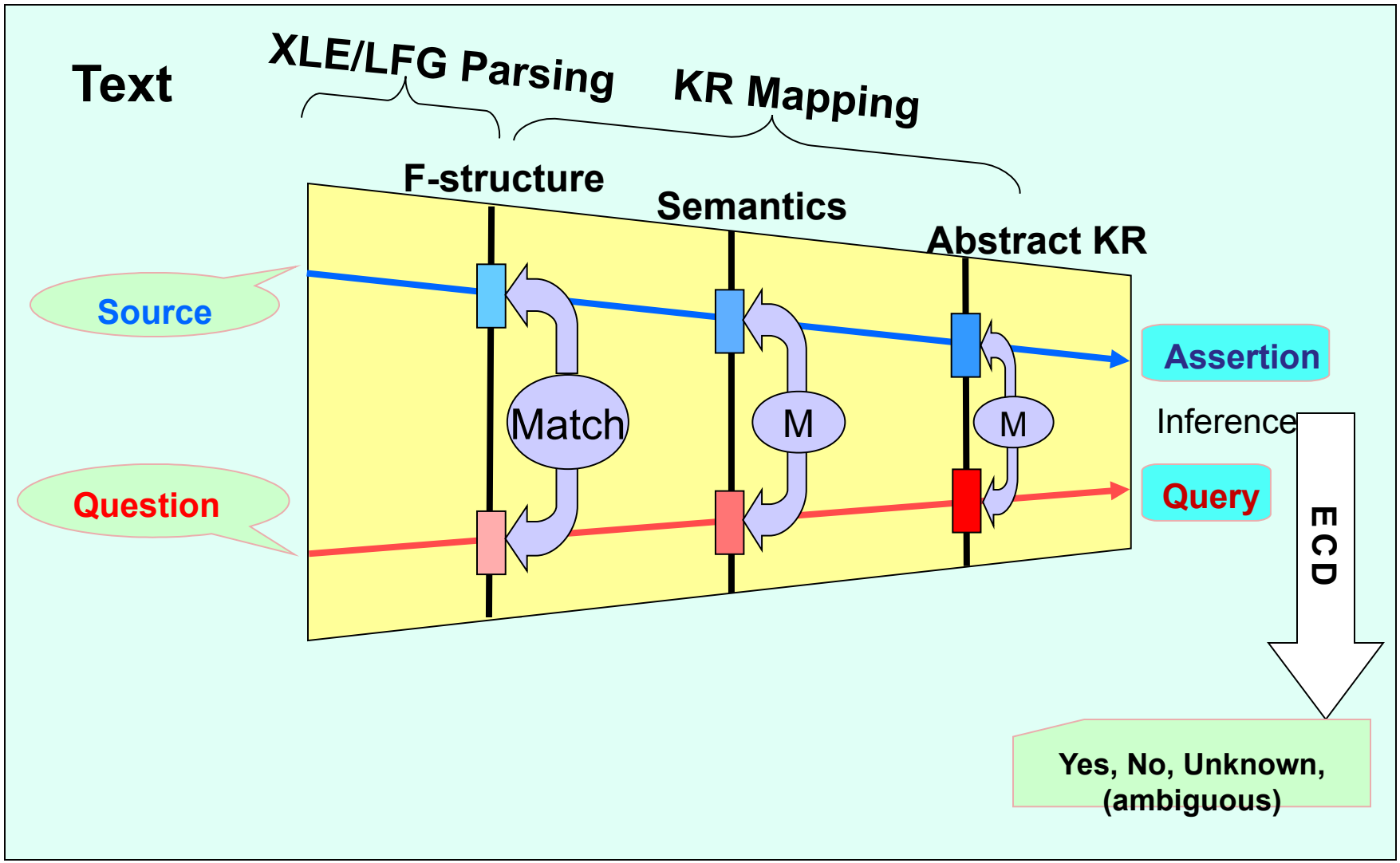
- Motivation: access control must be logic…
- Background
- Basic architecture and examples
- Discussion & further improvements

parc
Palo Alto Research Center

Why is every one thinking of Logics of Access Control?

- Ubiquity of computing and growth of the Internet turned Information Security into a central area of research in computer science.
- Every one is doing it.
- For logicians there's considerable amount of work on logical methods for Access Control
- For example:
  - Abadi et al, 1993, Abadi, 2003
  - Garg et al, 2006
  - Garg, Pfenning 2006
  - Garg, Abadi, 2008

ASKER architecture



Regression testing

- Regression is a way of testing that the system or a system component is improving
- Must check all components/layers of the system:
  - syntax
  - semantics
  - AKR
  - ECD algorithm
- Two kinds of regression testing
  - Construct testsuite of sentences and **bank** all representations (trees/f-structures, semantics, akrs, answers)
    - » compare current version against banked version
  - Construct testsuites of question-answer pairs with human answer
    - » compare system answer to human answer

How to do regression testing?

- Visually inspecting all the representations is hard.
- Checking when representations change is possible
  - need to check system improvement
- Checking pairs of question-answers where the system answer differs from the human answer is easier.
- Different types of testsuites
  - Hand-crafted and “in the wild” texts
  - Phenomena-based and mixed collections
  - **sanity checks** (self matching)

Architecture of Regression Testing

- Have development sets, capability sets, test sets, sanity checks.
- Development examples > bigger and more complicated
- Test examples > simpler and more focused
- Using qa (question-answer) pairs is **indirect** way of checking representations
- Determine which component needs improvement

parc
Palo Alto Research Center

Must check the basics don't get broken…

Sanity checks:

- an assertion P (for passage) should always entail the assertion P.
- given assertion P and the corresponding question P?, should always get Yes.
- given an assertion equivalent to P&Q this should always entail both P and Q.

Must check our basic analyses are not broken

basic development sets testsuites

- Deverbals (Rome's destruction of Carthage→ Rome destroyed Carthage)
- Factive/implicative verbs (Ed knows that Mary arrived→ Mary arrived)
- Factive nouns (The fact that the Earth is flat surprised Mary→ The Earth is flat --according to the author of the sentence)
- Quantifiers (Every man left→ Every tall man left)
- Copula, appositives, coordination, etc.
- Anaphora

Developing a regression system

Focus on what developers need from the testsuites to

- ensure systematic development
- track down and correct any loss of coverage
- identify areas for further development
- help multiple developers work simultaneously

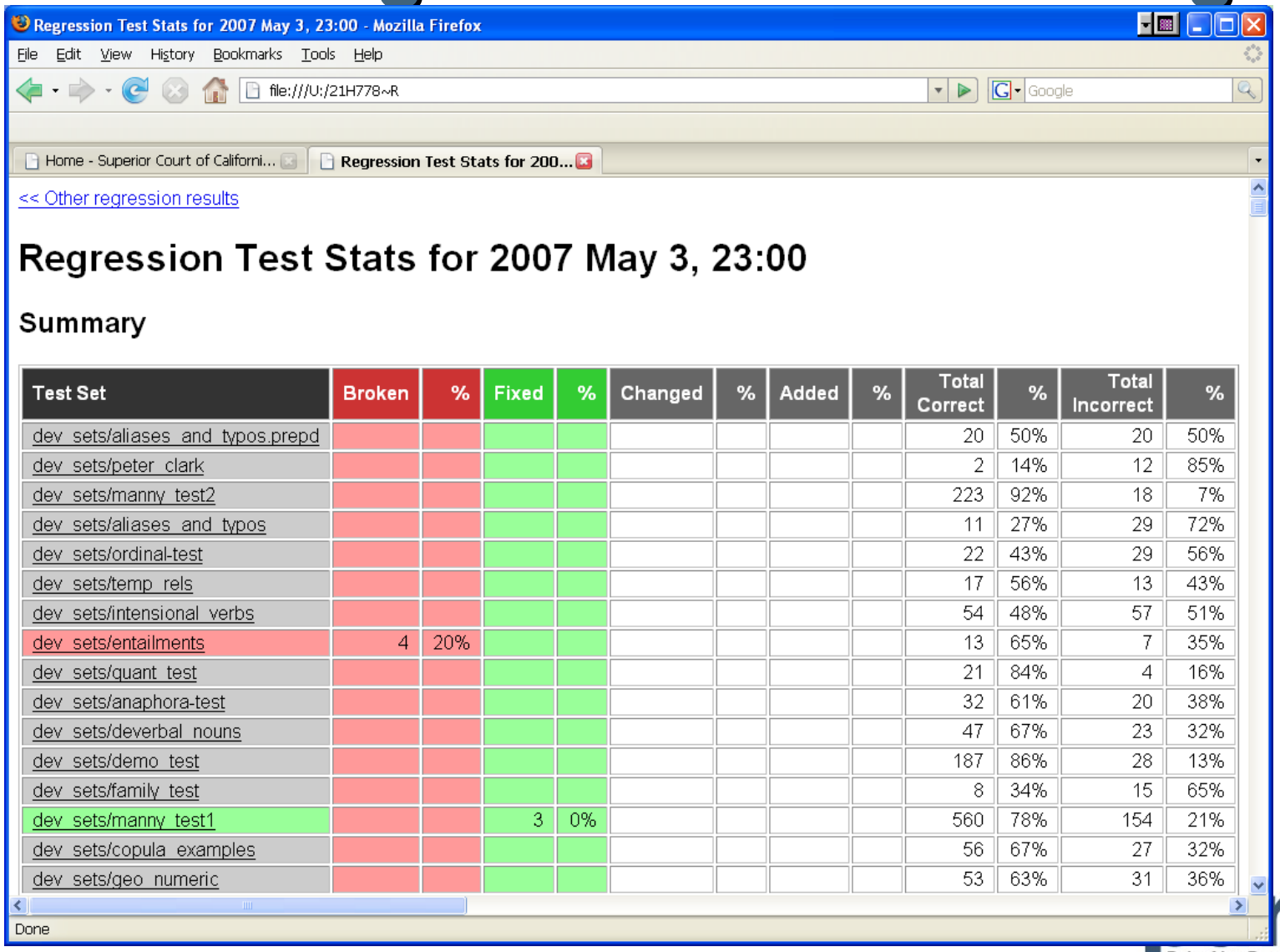
Using the regression system

- Regular system builds: daily/weekly
- Check-out code and grammars from CVS/ SVN repository
- Run regression tests
- Upload test summary, release to web-server and send email

Essential regression system features:

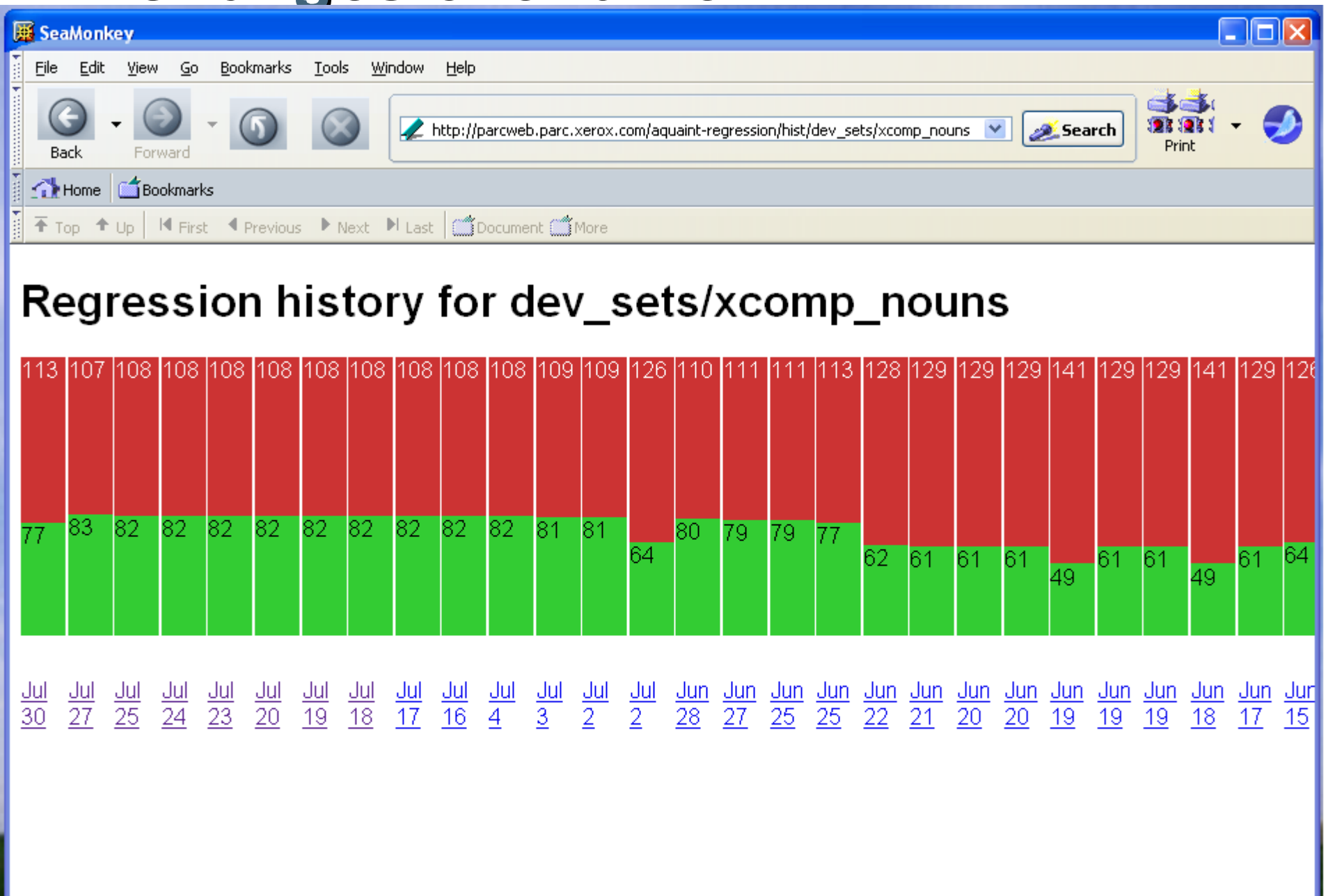
1. easy to run
2. clear presentation of results
3. methods to compare results with previous runs
4. serve different team members (grammar writers, semanticists, lexical resources creators, system integrators, etc).

Visualizing the results of testing



Test Set	Broken	%	Fixed	%	Changed	%	Added	%	Total Correct	%	Total Incorrect	%
dev_sets/aliases_and_typos.prepd									20	50%	20	50%
dev_sets/peter_clark									2	14%	12	85%
dev_sets/manny_test2									223	92%	18	7%
dev_sets/aliases_and_typos									11	27%	29	72%
dev_sets/ordinal-test									22	43%	29	56%
dev_sets/temp_rels									17	56%	13	43%
dev_sets/intensional_verbs									54	48%	57	51%
dev_sets/entailments	4	20%							13	65%	7	35%
dev_sets/quant_test									21	84%	4	16%
dev_sets/anaphora-test									32	61%	20	38%
dev_sets/deverbal_nouns									47	67%	23	32%
dev_sets/demo_test									187	86%	28	13%
dev_sets/family_test									8	34%	15	65%
dev_sets/manny_test1			3	0%					560	78%	154	21%
dev_sets/copula_examples									56	67%	27	32%
dev_sets/geo_numeric									53	63%	31	36%

Changes over time



Regression implementation summary

- Regression system
  - Check-out system from repository
  - Run regression tests
  - Release test summary to web-server and email
- Multiple regression suites
  - Component testing
  - End-to-end system

Lessons learned

- Regression testing should be
  - automated
  - nightly
- User interface matters
  - easily see what broke
  - comparison over time
- Testing of
  - units (syntax, semantics, akr)
  - entire system

Some (anecdotal) Stats

- AKR Regression started in Jan 2007
- 34 testsuites in dev_sets, 1 in capability sets (902 qas), 13 in test_sets and 20 in sanity.
- Approximately 16K qa pairs total, 10.3 K dev+cap+sanity.
- Average correction rate total 54%,

average on dev+ cap+ sanity = 79%

(don't have average on dev or dev+cap only– still sorting out sensible ways of separating results)

aiming for 100% on sanity, though…

parc
Palo Alto Research Center

Thank you

Questions?

parc
Palo Alto Research Center

Improvements on regression testing I

- More linguistic phenomena, more controlled interaction
- Gap between lab examples and in the wild ones too big?
- History gives numbers of correct and broken qas, not whether the correct ones are persistent
- Maybe can get a minimum core of totally correct qas, without which one should not commit a change?

Improvements on regression testing II

- Diff on the representations themselves?
- Time reports?
- Different kind of report for different kinds of testsuites? Sanity ones need to see which ones are *still* broken.
- Aggregated results by kind?
- Need a interpolation kind of theorem:

if P -> Q, then there exists P', (provable from P) such that P'=Q?

parc
Palo Alto Research Center

RTE3 Results

	Gold YES	Sys YES	Cor-rect	R	P	F
Strict ECD	410	31	25	6	84	22
Loose ECD	410	52	37	8,8	71	25

parc
Palo Alto Research Center

NYT 2000: top of the chart

be (un)able to	> 12600
fail to	> 5000
know that	> 4800
acknowledge that	> 2800
be too ADJ to	> 2400
happen to	> 2300
manage to	> 2300
admit/concede that	> 2000
be ADJ enough to	> 1500
have a/the chance to	> 1500
go on/proceed to	> 1000
have time/money to	> 1000

Robust software engineering approach to testing

- Must check the basics don't get broken…
- Regression tests running since January, 24.
- Long pipeline, multiple developers, need mechanisms to ensure system improves
- Several tools developed in parallel: wixcel (collaborative spreadsheets), svn, wiki, bugzilla, regression website, etc
- Sanity checks, histories more recent
- More about it later

The NLP-KR Gap

- There have been parallel efforts on text-based and knowledge-based question answering
  - Benefits of knowledge-based approaches inaccessible without some bridge from text to KR.

- Prime Goal
  - Robust, broad coverage mapping from texts and questions to KR
  - Allow KRR systems with lots of world knowledge to answer questions

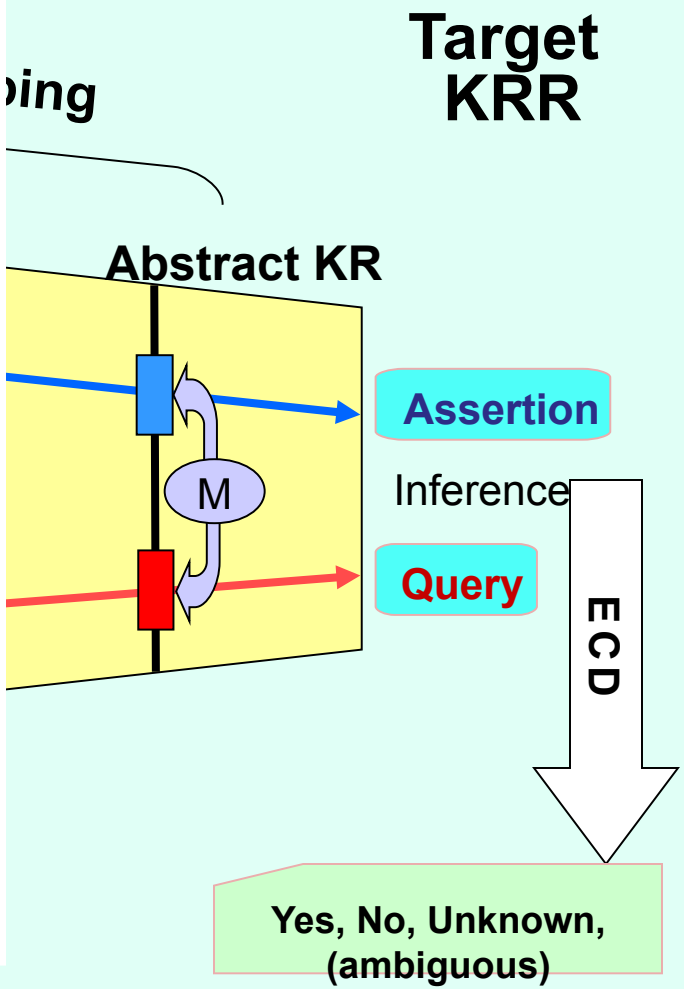
- Second-thoughts Goal
  - Robust, broad coverage mapping from texts and questions to KR
  - Allow system with basic knowledge of English to answer questions

System can do normalizations…

- Argument structure:
  - Mary bought an apple./An apple was bought by Mary.
- Interrogative/assertive forms
  - Mary bought an apple. Did Mary buy an apple?
- Synonyms and hypernyms:
  - Mary bought/purchased/acquired an apple.
- Factivity and contexts:
  - Mary managed/failed to buy an apple.
  - Ed prevented Mary from buying an apple.
  - We know/said/believe that Mary bought an apple.

ASKER User Interface screen shot?

- John arrived this morning. Did John arrive? Yes
- John didn't arrive. Did John arrive? No
- John arrived. Did John arrive this morning? Unknown
- John didn't wait to buy his apple. Did John buy his apple?



Must increase the phenomena we can deal with…

- Does {Mary did not buy an apple.} imply that {Mary bought an apple.}?

  Answer is clearly NO and the system can provide it.

- Does {Mary dislikes apples.} imply that {Mary likes apples}?
  - Answer is clearly NO, system can only produce UNKNOWN, need to add marking of antonyms to our Unified Lexicon.
- Does {Mary does her laundry.} imply that {Mary washes her clothes.}?

  Answer is YES, but system cannot deal with this kind of paraphrase, yet.

Regression system

Consists mainly of

1. building and maintaining test suites and
2. using these test suites to evaluate
   - Grammatical/semantic/KR coverage
   - accuracy of analysis & overgeneration
   - progress
   - detailed performance characteristics of parser/transfer/ECD (light inference) mechanism
   - the performance of the components as components and their interaction
   - how grammar/semantics/KR changes affect the whole system.