Augmenting Robot Knowledge Consultants with Distributed Short Term Memory

Tom Williams[1], Ravenna Thielstrom[2], Evan Krause[2], Bradley Oosterveld[2], and Matthias Scheutz[2]

[1] Colorado School of Mines MIRRORLab, Golden, CO, USA
`twilliams@mines.edu`, mirrorlab.mines.edu
[2] Tufts University Human-Robot Interaction Lab, Medford, MA, USA
{`firstname.lastname`}`@tufts.edu`, hrilab.tufts.edu

Abstract. Human-robot communication in situated environments involves a complex interplay between knowledge representations across a wide variety of modalities. Crucially, linguistic information must be associated with representations of objects, locations, people, and goals, which may be represented in very different ways. In previous work, we developed a Consultant Framework that facilitates modality-agnostic access to information distributed across a set of heterogeneously represented knowledge sources. In this work, we draw inspiration from cognitive science to augment these distributed knowledge sources with Short Term Memory Buffers to create an STM-augmented algorithm for referring expression generation. We then discuss the potential performance benefits of this approach and insights from cognitive science that may inform future refinements in the design of our approach.

Keywords: Natural language generation, working memory, cognitive architectures

1 Introduction

Social robots engaging in natural task-based dialogues with human teammates must understand and generate natural language expressions that refer to entities such as people, locations, and objects [18, 23]. These tasks, known as *reference resolution* and *referring expression generation (REG)*, are particularly challenging in realistic robotics applications due to the realities of how knowledge is represented and distributed in modern robotic architectures.

We previously presented a *Consultant Framework* [35] that allows a robot to use its distributed sources of knowledge during reference resolution [39] and REG [40], without requiring the language processing system to understand how that knowledge is represented and accessed. We've used this framework in previous work to enable a modern take on the classic Incremental Algorithm (IA) [9] for REG, relaxing several assumptions: that knowledge is certain, that knowledge is centrally stored, and that a list of all properties known to hold for each known entity is centrally available during REG. Our Consultant Framework allows these assumptions to be relaxed, producing a REG algorithm tailored to the realities of robotic architectures. Domain independence, however, comes at a computational cost, especially for language generation.

The IA requires, for each property that could be included, consideration of whether it holds for the to-be-described target and not for at least one distractor. Under the assumptions of the IA, this can be performed via set-membership checks on centrally available property sets. When the assumption of such property sets is relaxed, however, as is the case in the modified algorithm designed to leverage our Consultant Framework, these considerations must instead be made through queries to the Consultants responsible for the target and distractors. The computational complexity of REG combined with the computational cost of these queries results in a significant computational burden.

To address this computational burden, we propose(see also [38]) an augmented Consultant Framework that includes Consultant-Specific Short Term Memory (STM) Buffers that cache a small number of properties recently determined to hold for various entities. We will begin by defining this augmented framework, and by describing how it reduces the complexity of REG. We will then discuss different possible assumptions that can be made in the design of these STM Buffers, and discuss how these different choices impact both efficiency and cognitive plausibility. Next, we will discuss insights that can be gleaned from psychological models of memory decay and forgetting and computational caching strategies, and how these insights apply to the design of these STM Buffers. Finally, we will discuss how these Buffers can be exploited by processes beyond REG, and their potential relation to other cognitive models maintained throughout the architecture.

2 Augmented Framework

Our previously presented Consultant Framework [40] allows information about entities to be assessed when knowledge is uncertain, heterogeneous, and distributed, facilitating IA-inspired approaches to REG. Specifically, each Consultant c facilitates access to one KB k, and must be capable of four functions:

1. providing a set c_{domain} of atomic entities from k,
2. advertising a list $c_{constraints}$ of constraints that can be assessed with respect to entities from c_{domain}, *and that is ordered by descending preference.*
3. assessing constraints from $c_{constraints}$ with respect to entities from c_{domain}, and
4. adding, removing, or imposing constraints from $c_{constraints}$ on entities from c_{domain}.

In this section, we define an *(STM)-Augmented Consultant Framework* that adds an additional requirement:

5. providing a list c_{STM} of properties that hold for some entity from c_{domain}.

Crucially, the properties returned through this capability do not need to be all of the properties that hold for the target entity. A Consultant may have a large number of properties that it could assess for a given entity if need be, some of which might be very expensive to compute. As such, the purpose of this capability is not to request evaluation of all possible properties for the specified entity, but rather to request the contents of a small cache of properties recently determined to hold for the specified entity.

Models of Working Memory suggest that humans maintain cached knowledge of a small number of activated entities. While early models of working memory suggested that the size of working memory is bounded to a limited *number of chunks* (as in Miller's famous "magical number" of seven, plus or minus two [19]), more recent models instead suggest that the size of working memory is affected by the complexity of those chunks [17]. For example, needing to maintain multiple *features* of a single entity may detract from the total number of maintainable entities, and accordingly, the number of features maintainable for other entities [1, 31, 20]. Moreover, recent research has suggested that humans may have different resource limits for different *types* of representations (e.g., visual vs. auditory [11], or different types of visual features [34]) either due to the existence of separate domain-specific cognitive resources [2, 15] or do to decreased interference between disparate representations [22].

Drawing on these insights, the new capability required in the *STM-Augmented Consultant Framework* requires each Consultant to maintain its own set of *features* currently remembered for the set of entities for which it is responsible. This serves to allow fast access to a set of entity properties likely to be relevant, in order to avoid the expensive long-term memory queries that make processes such as referring expression generation so expensive in the current Consultant framework. In the next section we describe how our newly proposed framework can be used during the course of REG.

3 Algorithmic Approach

We now present *SD-PIA*, the STM-Augmented variant of the Distributed, Probabilistic IA [40]. We will describe the main differences between *DIST-PIA* and *SD-PIA*, the key difference being our use of the properties stored in STM before performing LTM-Query intensive operations. While DIST-PIA crafted sub-descriptions through the use of a single algorithm (DIST-PIA-H), *SD-PIA* begins by crafting an initially (possibly partial) sub-description using the *SD-PIA-STM-H* algorithm, which utilizes only the properties found in STM Buffers. If the sub-descriptions returned through this algorithm are not fully discriminating, the partial sub-description is augmented by passing the set of still-to-be-eliminated distractors to *SD-PIA-H*, which operates much the same as the original *DIST-PIA-H* algorithm.

The other main difference between *DIST-PIA* and *SD-PIA* is in the design of the *SD-PIA-STM-H* function. Instead of considering all properties advertised in the target's domain, *SD-PIA-STM-H* considers only the properties returned by querying that Consultant's STM buffer, requiring a single query rather than $O(c_m^A)$ queries. For each of these already-known-to-hold and already-bound queries, *SD-PIA-STM-H* iteratively rebinds the query to refer to each distractor x rather than the target entity. For each re-bound query, *SD-PIA-STM-H* calls a function *stm-apply*, which checks whether that property holds for that distractor

(x), by first checking whether the property exists in the STM Buffer maintained by Consultant c_x for x, or, if and only if this is not the case, by checking whether the property is known to hold by Consultant c_x using its' *apply* method, as usual.

Notation

C	A set of *Consultants* $\{c_0, \ldots, c_n\}$
c_m^Λ	The set of formulae $\{\lambda_0, \ldots, \lambda_n\}$ advertised by Consultant c responsible for m.
$c_m^{\Lambda STM}$	The STM buffer of formulae maintained by Consultant c responsible for m.
M	A robot's *world model* of entities $\{m_0 \ldots m_n\}$ found in the domains provided by C.
D	The incrementally built up description, comprised of mappings from entities M to sets of pairs (λ, Γ) of formulae and bindings for those formulae.
D^M	The set of entities $m \in M$ for which sub-descriptions have been created.
d^M	The set of entities $m \in M$ involved in sub-description d.
P	The set of candidate (λ, Γ) pairs under consideration for inclusion.
Q	The queue of referents which must be described.
X	The incrementally pruned set of distractors

Algorithm 1 *SD-PIA*(m, C)

```
1: $D$ = new Map() // The Description
2: $Q$ = new Queue($m$) // The Referent Queue
3: while $Q \neq \emptyset$ do
4:     // Consider the next referent
5:     $m\prime$ = pop(Q)
6:     // Craft a description d for it
7:     $(d, X)$ = SD-PIA-STM-H$(m\prime, C)$
8:     $d$ = SD-PIA-H$(m\prime, C, X, d)$
9:     $D = D \cup \{m \rightarrow d\}$
10:    // Find all entities used in d
11:    for all $m\prime\prime \in d^M \setminus keys(D)$ do
12:        // And add undescribed entities to the queue
13:        push$(Q, m\prime\prime)$
14:    end for
15: end while
16: return $D$
```

4 Demonstration

We will now present a proof-of-concept demonstration of our proposed algorithm, implemented in the ADE [27] implementation of the DIARC architecture [29, 25].The ADE (Agent Development Environment) middleware provides a well-validated infrastructure for enabling agent architectures through parallel distributed processing. The Distributed Integrated Affect Reflection Cognition (DIARC) Architecture is a component-based architecture that has been under development for over 15 years, with a focus on robust spoken language understanding and generation. For our demonstration scenario, the following architectural components were used: Speech Recognition (using the Sphinx4 Speech

Algorithm 2 *SD-PIA-STM-H*(m, C)

```
1: $d = \emptyset$ // The Sub-Description
2: $X = M \setminus m$ // The Distractors
3: $P = order([\forall \lambda \in c_m^{\Lambda_{STM}} : (\lambda, \emptyset)], c_m^{\Lambda})$
4: while $X \neq \emptyset$ and $P \neq \emptyset$ do
5:     $(\lambda, \Gamma) = pop(P)$
6:     $\bar{X} = [x \in X \mid stm\_apply(c_x, \lambda, rebind(\Gamma, m \rightarrow x)) < \tau_{dph}]$
7:     if $\bar{X} \neq \emptyset$ then
8:         $d = d \cup (\lambda, \Gamma)$
9:         $X = X \setminus \bar{X}$
10:    end if
11: end while
12: return $(d, X)$
```

Algorithm 3 *SD-PIA-H*(m, C, X, d)

```
1: // Initialize a set of properties to consider: those advertised by the Consultant c responsible
   for m and not already part of the sub-description
2: $P = [\forall \lambda \in c_m^{\Lambda} : (\lambda, \emptyset)] \setminus d$
3: // While there are distractors to eliminate or properties to consider
4: while $X \neq \emptyset$ and $P \neq \emptyset$ do
5:     $(\lambda, \Gamma) = pop(P)$
6:     // Find all unbound variables in the next property
7:     $V = find\_unbound(\lambda, \Gamma)$
8:     if $|V| > 1$ then
9:         // If there's more than one, create copies under all possible variable bindings that leave
           one variable of the same type as the target unbound
10:        for all $\Gamma\prime \in cross\_bindings(\lambda, \Gamma, C)$ do
11:            // And push them onto the property list
12:            $push(P, (\lambda, \Gamma\prime))$
13:        end for
14:        // Otherwise, if sufficiently probable that the property applies to the target...
15:    else if $apply(c_m, \lambda, \Gamma \cup (v_0 \rightarrow m)) > \tau_{dph}$ then
16:        // And sufficiently probable that it does not apply to at least one distractor...
17:        $\bar{X} = [x \in X \mid apply(c_x, \lambda, \Gamma \cup (v_0 \rightarrow x)) < \tau_{dph}]$
18:        // Then bind its free variable to the target, and add it to the sub-description...
19:        if $\bar{X} \neq \emptyset$ then
20:            // And remove any eliminated distractors
21:            $d = d \cup (\lambda, \Gamma \cup (v_0 \rightarrow m))$
22:            $X = X \setminus \bar{X}$
23:        end if
24:    end if
25: end while
26: return $d$
```

Recognizer [33]), Parsing (which uses the most recent iteration [28] of the TLDL Parser [10]), the Dialogue and Pragmatics Components [6],the NLG Component (in which the *SD-PIA* algorithm was implemented), the Goal Manager [26], the Belief Component (which provides a Prolog Knowledge Base [8]), the Resolver Component [39],the GROWLER HyperResolver Component [37], and a simulated Vision Component [14] (which serves as a Consultant).

For this demonstration walkthrough, we use a simple scenario involving a single "objects Consultant" (the aforementioned simulated Vision Component), which advertises a variety of constraints, e.g., related to object type and object color, with type constraints having higher preference than color constraints and ordered according to specificity. The scene in front of the robot contains a red teabox (known to the objects Consultant as $object_1$) and a green teabox (known to the objects Consultant as $object_2$) sitting on a table (known to the objects Consultant as $object_3$).

For this walkthrough, we begin by instructing the simulated robot "Look at the box". The TLDL Parser [28, 10] parses this into an utterance of type *Instruction*, with primary semantics $lookat(self, X)$ and supplemental semantics $\{box(X)\}$. The GROWLER reference resolution algorithm described in our recent work [37] (see also [36, 42]), then identifies the two teaboxes ($object_1, object_2$), as candidate referents satisfying the given description. During the reference resolution process, when the property $box(X)$ is determined to hold for each entity, it is placed into that entity's STM Buffer within the simulated Vision Component. Because the expression is ambiguous, a clarification request is automatically generated [41] to determine whether $object_1$ or $object_2$ is the target referent. For each of these entities, *SD-PIA* is recruited to generate referring expressions. In this section, we will describe the process followed in the selection of properties for $object_1$ alone (hereafter o_1), as the process for $object_2$ is identical in structure.

SD-PIA begins by creating empty description $D = \emptyset$ and referent queue $Q = \{o_1\}$ (Alg. 1, Lines 1- 2). Because there are still referents to describe (Line 5), *SD-PIA* calls helper function *SD-PIA-STM-H* (*STM-H* hereafter) to craft a sub-description for o_1, which is popped off of Q (Line 7).

STM-H begins by asking the Consultant responsible for o_1 for a set of distractors X (e.g., $\{o_2, o_3\}$) and the set of properties P stored in its STM Buffer for o_1, sorted by that Consultant's preference ordering c^{Λ}_{m}. (Alg. 2 Lines 2-3), in this case $box(X)$.Next, *STM-H* constructs the reduced set of distractors $\bar{X}$ for which the property does *not* appear in STM (i.e., o_3). Because this is nonempty, o_3 is removed from the set of distractors, and $box(o_1)$ is added to sub-description d (Alg. 2 Lines 8- 9). Because there are no more properties to examine in P, sub-description d and the set of remaining distractors $\{o_2\}$ is returned.

The second helper function, *SD-PIA-H* (Alg. 3, hereafter simply *HELPER*), is then used to complete the referring expression. *HELPER* begins by asking the Consultant responsible for o_1 for the set of properties P it can use in descriptions, sorted according to its predetermined preference ordering, and ignoring properties already contained in sub-description d, e.g., teabox(X), table(X), red(X), green(X), on(X,Y) (Alg. 3 Line 2).

From this list, *HELPER* pops the first unconsidered property (i.e., $teabox(X)$) and its (empty) set of bindings. $teabox(X)$ has exactly one unbound variable, so *HELPER* will use Consultant o's *apply* method (as per Capability 3) to ask how probable it is that $teabox(X)$ applies to o_1 (Alg. 3 Line 15). Because it is sufficiently probable that this property applies to this entity, *HELPER* uses the same method to determine whether it also applies to the single remaining distractor (o_2). Because it does, the property will be ignored.

HELPER will then repeat this process with other properties. Suppose it is insufficiently probable that $table(X)$ holds: it will be ignored. Suppose it *is* sufficiently probable that $red(X)$ holds but not for the lone remaining distractor (o_2), allowing o_2 to be ruled out. Thus, $\{o_2\}$ will be removed from X,and $red(o_1)$ will be added to d.Since X is now empty, the sub-description $\{(box(o_1), red(o_1)\}$

will be returned to *SD-PIA* (Line 26) and added to full description D. Since this sub-description does not refer to any entities that have yet to be described and Q is empty, *SD-PIA* will return description D (Algorithm 1, Line 16). This process is then repeated for object o_2. It will be the responsibility of the next component of the natural language pipeline to translate this into an RE, e.g. “Do you mean the red box or the green box?” [41].

5 Discussion

Potential Benefits: The primary motivation behind our approach is performance: the number of queries needed when determining what properties to use may be much lower when those properties are sufficiently discriminating. Similarly, determining whether properties rule out distractors may be possible using set-membership checks rather than costly long term memory queries. Moreover, we believe these buffers may facilitate *lexical entrainment* [4, 5], where conversational partners converge on common choices of labels and properties over the course of a conversation, resulting in more comprehensible referring expressions [32]. If a robot’s STM Buffers are populated with properties used by itself and its interlocutors, and if the properties contained in those buffers are considered before others, this may directly lead to such entrainment.
Potential Limitations: On the other hand, because the robot arbitrarily restricts itself to a subset of the properties it *could* otherwise choose to use, it may force the robot into local maxima in the landscape of referring expressions. Moreover, the robot runs the risk of using a property that does not actually hold if it does not appropriately handle contextual dynamics. For example, an object previously described as “on the left” may no longer be “on the left” if the object, robot, or interlocutor has moved since the object was last discussed.
Design Decisions: Buffer Size Limitations: Many insights from psychology could be leveraged to prevent such mistakes. A context-sensitive decay-based model of working memory might prevent this by having different properties “decay” out of cache, with time or probability of decay proportional to how likely it is to change over time [3, 30]. A resource-based model might prevent this by having a limited total buffer size, and have property dynamics factor into the decision of what to bump from memory when new things need to be inserted into an already-full buffer [13, 16].Finally, an interference-based model might prevent this by having properties added to a buffer “overwrite” the most similar property currently in the buffer [21, 24]. These are loose characterizations of the respective theories from cognitive psychology; a comprehensive discussion of these theories and the relative evidence for them from a psychological perspective can be found in [20]. Of course, the approach taken need not be cognitively plausible. The robot could, for example, statistically model the dynamics of different properties, and use them to periodically re-sample the properties held in its buffers.

The question of cognitive plausibility also raises a different question: how extensive should the robot’s memory caches be? Should the robot keep property caches for all entities, for only those that are relevant in the current context,

or for an even smaller set? And for each entity, should the robot track all relevant knowledge for so long as that entity is tracked, or should it track only a fixed, small number of properties? And should such limits be local, or global limits shared between tracked entities? These are once again questions for which candidate answers can be gleaned from the psychological literature [20]. Here, interesting tradeoffs can be made. On the one hand, robots can be made to remember much more than humans can. On the other hand, expanded memory may come at a computational cost; and moreover, choosing to remember more means increased risk of incorrect behavior due to mishandling of property dynamics. Further evidence from neuroimaging studies suggests that the contents of working memory in humans is biased by humans' current goals, with only task-relevant features being maintained in visual working memory [43]. For robots, task-relevance may be similarly useful for optimally selecting what to maintain in Consultants' STM Buffers.
Design Decisions: Buffer Population: Similar tradeoffs arise when deciding when to add properties used in natural language to a robot's STM Buffers. In this work, properties are added to buffers as soon as they are determined to hold for a referent, rather than waiting until the end of the reference resolution process. This means that lexical entrainment effects may be seen for entities other than intended referents. For example, consider "the tall red box". Assuming properties are processed in the order $tall(X)$, $red(x)$, $box(X)$, all tall objects in the scene will have $tall(X)$ added to their STM Buffers, all tall red objects will also have $red(X)$ added to their STM Buffers, and tall red boxes will have all three properties added to their STM Buffers. Accordingly, the robot will prefer to use height and potentially color and shape to refer to other objectseven if they have not been referred to. This is not dissimilar from psycholinguistic observationsof syntactic and semantic carry-over from previous object references to references to previously unmentioned entities [12, 7].This could be particularly effective in the case of the Vision Consultant, as additional caching reduces the risk of needing to conduct future (potentially expensive) visual searches. If this particular memory-performance tradeoff is not optimal for Consultants associated with non-visual modalities, the decision to add properties to STM Buffers may need to be made on a per-Consultant basis.

6 Conclusions

We have presented a caching strategy augmenting a robot's set of distributed knowledge sources with cognitively inspired STM Buffers so as to increase computational efficiency. We have explained how these buffers may facilitate linguistic phenomena such as lexical entrainment, and identified insights from cognitive science that may inform future refinements of these Buffers. Two tasks will be crucial for future research building off this work. First, the presented approach must be evaluated both in terms of computational performance and with respect to objective and subjective measures such as those used in our previously presented evaluative approach [40]. Second, insights from cognitive science must be leveraged to enable alternate design decisions that may be explored both from the perspectives of artificial intelligence for human-robot interaction, cognitive modeling, and cognitive architecture.

Acknowledgments

This work was in part supported by ONR grant N00014-16-1-0278.

References

1. Alvarez, G.A., Cavanagh, P.: The capacity of visual short-term memory is set both by visual information load and by number of objects. Psychological science (2004)
2. Baddeley, A.: Working memory. Science **255**(5044), 556–559 (1992)
3. Baddeley, A.D., Thomson, N., Buchanan, M.: Word length and the structure of short-term memory. Journal of verbal learning and verbal behavior (1975)
4. Brennan, S.E.: Lexical entrainment in spontaneous dialog. ISSD **96**, 41–44 (1996)
5. Brennan, S.E., Clark, H.H.: Conceptual pacts and lexical choice in conversation. Journal of Experimental Psychology: Learning, Memory, and Cognition (1996)
6. Briggs, G., Scheutz, M.: A hybrid architectural approach to understanding and appropriately generating indirect speech acts. In: Proc. AAAI (2013)
7. Carbary, K., Tanenhaus, M.: Conceptual pacts, syntactic priming, and referential form. In: Proc. CogSci Workshop on Production of Referring Expressions (2011)
8. Clocksin, W., Mellish, C.S.: Programming in PROLOG. Springer Sci.&Bus. (2003)
9. Dale, R., Reiter, E.: Computational interpretations of the gricean maxims in the generation of referring expressions. Cognitive Science **19**(2) (1995)
10. Dzifcak, J., Scheutz, M., Baral, C., Schermerhorn, P.: What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In: Proc. ICRA (2009)
11. Fougnie, D., Zughni, S., Godwin, D., Marois, R.: Working memory storage is intrinsically domain specific. Journal of Experimental Psychology: General (2015)
12. Goudbeek, M., Krahmer, E.: Alignment in interactive reference production: Content planning, modifier ordering, and referential overspecification. TiCS (2012)
13. Just, M.A., Carpenter, P.A.: A capacity theory of comprehension: individual differences in working memory. Psychological review **99**(1), 122 (1992)
14. Krause, E., Zillich, M., Williams, T., Scheutz, M.: Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. In: Proc. of the Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
15. Logie, R.H.: Visuo-spatial working memory. Psychology Press (2014)
16. Ma, W.J., Husain, M., Bays, P.M.: Changing concepts of working memory. Nature neuroscience **17**(3), 347 (2014)
17. Mathy, F., Feldman, J.: What's magic about magic numbers? chunking and data compression in short-term memory. Cognition **122**(3) (2012)
18. Mavridis, N.: A review of verbal and non-verbal human–robot interactive communication. Robotics and Autonomous Systems **63**, 22–35 (2015)
19. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological review (1956)
20. Oberauer, K., Farrell, S., Jarrold, C., Lewandowsky, S.: What limits working memory capacity? Psychological bulletin (2016)
21. Oberauer, K., Kliegl, R.: A formal model of capacity limits in working memory. Journal of Memory and Language **55**(4), 601–626 (2006)

22. Oberauer, K., Lewandowsky, S., Farrell, S., Jarrold, C., Greaves, M.: Modeling working memory: An interference model of complex span. Psyc. Bull&Rev. (2012)
23. Popescu-Belis, A., Robba, I., Sabah, G.: Reference resolution beyond coreference: a conceptual frame and its application. In: Proc. COLING (1998)
24. Saito, S., Miyake, A.: On the nature of forgetting and the processing–storage relationship in reading span performance. Jour. Memory & Language (2004)
25. Schermerhorn, P.W., Kramer, J.F., Middendorff, C., Scheutz, M.: DIARC: A testbed for natural human-robot interaction. In: Proc. AAAI (2006)
26. Schermerhorn, P.W., Scheutz, M.: The utility of affect in the selection of actions and goals under real-world constraints. In: IC-AI. pp. 948–853 (2009)
27. Scheutz, M.: Ade: Steps toward a distributed development and runtime environment for complex robotic agent architectures. Applied Artificial Intelligence (2006)
28. Scheutz, M., Krause, E., Oosterveld, B., Frasca, T., Platt, R.: Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In: 16th Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2017)
29. Scheutz, M., Williams, T., Krause, E., Oosterveld, B., Sarathy, V., Frasca, T.: An overview of the distributed integrated cognition affect and reflection DIARC architecture. In: Cognitive Architectures (2018 (in press))
30. Schweickert, R., Boruff, B.: Short-term memory capacity: Magic number or magic spell? Jour. Exp. Psych.: Learning, Memory, and Cognition (1986)
31. Taylor, R., Thomson, H., Sutton, D., Donkin, C.: Does working memory have a single capacity limit? Jour. Memory & Language (2017)
32. Tolins, J., Zeamer, C., Fox Tree, J.E.: Overhearing dialogues and monologues: How does entrainment lead to more comprehensible referring expressions? Discourse Processes pp. 1–21 (2017)
33. Walker, W., Lamere, P., Kwok, P., et al.: Sphinx-4: A flexible open source framework for speech recognition. Tech. rep. (2004)
34. Wang, B., Cao, X., Theeuwes, J., Olivers, C.N., Wang, Z.: Separate capacities for storing different features in visual working memory. J. Exp. Psych. (2017)
35. Williams, T.: A consultant framework for natural language processing in integrated robot architectures. IEEE Intelligent Informatics Bulletin (2017)
36. Williams, T., Acharya, S., Schreitter, S., Scheutz, M.: Situated open world reference resolution for human-robot dialogue. In: Proc. HRI (2016)
37. Williams, T., Krause, E., Oosterveld, B., Scheutz, M.: Towards givenness and relevance-theoretic open world reference resolution. In: RSS Workshop on Models and Representations for Natural Human-Robot Communication (2018)
38. Williams, T., Krause, E., Oosterveld, B., Thielstrom, R., Scheutz, M.: Towards robot knowledge consultants augmented with distributed short term memory. In: RSS Workshop on Models and Reps. for Natural Human-Robot Comm. (2018)
39. Williams, T., Scheutz, M.: A framework for resolving open-world referential expressions in distributed heterogeneous knowledge bases. In: Proc. AAAI (2016)
40. Williams, T., Scheutz, M.: Referring expression generation under uncertainty: Algorithm and evaluation framework. In: Proceedings of the 10th International Conference on Natural Language Generation (INLG) (2017)
41. Williams, T., Scheutz, M.: Resolution of referential ambiguity in human-robot dialogue using Dempster-Shafer theoretic pragmatics. In: Proc. RSS (2017)
42. Williams, T., Scheutz, M.: Reference in robotics: A givenness hierarchy theoretic approach. In: The Oxford Handbook of Reference (2018 (in press))
43. Yu, Q., Shim, W.: Occipital, parietal, and frontal cortices selectively maintain task-relevant features of multi-feat. objects in visual working memory (2017)