Negation (short-form) in Korean is done by placing the negator, \(an(i)\), before the predicate.

(1) a. eysute-ka ka-n-ta. 
   Esther-NOM go-PRES-DECL
   ‘Esther is going.’

   b. eysute-nun sengsilha-ta. 
   Esther-TOP sincere-DECL
   ‘Esther is sincere.’

(2) a. eysute-ka \(an(i)\) ka-n-ta. 
   Esther-NOM NEG go-PRES-DECL
   ‘Esther is not going.’

   b. eysute-nun \(an(i)\) sengsilha-ta. 
   Esther-TOP NEG sincere-DECL
   ‘Esther is not sincere.’

Negator distribution (3) \((cf.\) negative prefixes (4)), negative polarity item licensing (5), and scope interactions with quantifiers (6) show that short-form negation is syntactic.

(3) a. *eysute-ka \(an(i)\) \(an(i)\) ka-n-ta. 
   Esther-NOM NEG NEG go-PRES-DECL
   ‘The book was not unpublished.’

   b. *eysute-nun \(an(i)\) \(an(i)\) sengsilha-ta.
   Esther-TOP NEG NEG sincere-DECL
   ‘John is not insincere.’

(4) a. chayk-i \(an(i)\) mikanhayngtoy-ess-ta.
   book-NOM NEG yet-publish-PAST-DECL
   ‘I didn’t meet anybody.’

   b. con-un \(an(i)\) pulsengsilha-ta.
   John-TOP NEG insincere-DECL
   ‘I didn’t meet anybody.’

(5) a. eysute-nun \(amu-to\) \(an(i)\) manna-ss-ta.
   Esther-TOP any-NPI NEG meet-PAST-DECL
   ‘I didn’t know any answer.’

   b. *eysute-nun \(amu-to\) manna-ss-ta.
   Esther-TOP any-NPI meet-PAST-DECL
   ‘I didn’t meet anybody.’

(6) a. eysute-nun \(motun\) chinkwu-lul \(an(i)\) manna-ss-ta. 
   Esther-TOP all friend-ACC NEG meet-PAST-DECL
   ‘All friends didn’t meet anybody.’

   b. \(motun\) haksayng-i \(an(i)\) sengsilha-ta.
   \(\forall\) > Neg: ‘not all students’

   all student-NOM NEG sincere-DECL
   ‘All students didn’t meet anybody.’

Two predicates show a suppletive negative form not related to the affirmative counterpart in form \((al-\ ‘know’ \rightarrow molu- ‘not know’ \((*an(i) al-); iss- ‘exist/present’ \rightarrow eps- ‘not exist/present’ \((*an(i) iss-))\). The suppletive negative predicates behave just like short-form negation cases, without the responsible negator.

(7) a. *eysute-ka na-lul \(an(i)\) molu-n-ta. 
   Esther-NOM I-ACC NEG know-PRES-DECL
   ‘I didn’t know any answer.’

   b. *cha-ka pakk-ey \(an(i)\) eps-ta.
   car-NOM rice-ACC NEG not.present-DECL
   ‘I didn’t know any answer.’

(8) a. na-nun \(amu\) tap-to moll-ass-ta. 
   I-TOP any answer-NPI not.know-PAST-DECL
   ‘I didn’t know any answer.’

   b. *na-nun \(amu\) tap-to al-ess-ta.
   I-TOP any answer-NPI know-PAST-DECL
   ‘I didn’t know any answer.’

(9) a. motun haksayng-i tap-ul molu-n-ta. 
   \(\forall\) > Neg: ‘not all students’

   all student-NOM answer-ACC not.know-PRES-DECL
   ‘Not all students didn’t know anything.’

   b. motun haksayng-i eps-ta. 
   \(\forall\) > Neg: ‘not all students’

   all student-NOM not.present-DECL
   ‘Not all students didn’t know anything.’

This paper proposes an analysis of suppletive negation in the framework of Distributed Mor-
The analysis is morphological fusion that takes two terminal nodes (negation and the predicate node under consideration) and turns them into one single node that contains all the morpho-syntactic features of the two original nodes. Fusion occurs in the postsyntactic morphological component within PF after overt syntax. In syntax, the regular short-form negation constructions and the suppletive negation constructions are identical in configuration, because nodes consist of morphological, syntactic and semantic features in syntax free of phonological features.

Assuming that predicate roots undergo head movement to C adjoining the intervening functional heads, the morphosyntactic structure of *molu-n-ta ‘not.know-PRES-DECL’ prior to vocabulary insertion looks like (10), which is structurally identical to the regular short-form negations like *an(i) manna-n-ta ‘not.meet-PRES-DECL’ in (11). The fusion operation is illustrated in (12). Then, vocabulary insertion applies to the fused node, with the vocabulary items in (13).

\((10) \ [^{\text{C}} \ [^{\text{T}} [^{\text{Neg}} [^{\text{+neg}} [^{\text{KNOW}} \text{Neg}] [^{\text{+pres}} \text{T}] [^{\text{+decl}} \text{C}]]]^{\text{T}}]^{\text{Neg}}] [^{\text{+neg}}]^{\text{KNOW}} \text{Neg} \] \rightarrow \ [^{\text{Neg}} [^{\text{+neg}} [^{\text{KNOW}} \text{Neg}]^{\text{+neg, KNOW}} \text{Neg}]^{\text{+decl}} \text{C}]

\((11) \ [^{\text{C}} \ [^{\text{T}} [^{\text{Neg}} [^{\text{+neg}} [^{\text{MEET}} \text{Neg}] [^{\text{+pres}} \text{T}] [^{\text{+decl}} \text{C}]]]^{\text{T}}]^{\text{Neg}}] [^{\text{+neg}}]^{\text{MEET}} \text{Neg} \] \rightarrow \ [^{\text{Neg}} [^{\text{+neg}} [^{\text{MEET}} \text{Neg}]^{\text{+neg, MEET}} \text{Neg}]^{\text{+decl}} \text{C}]

\((12) \ [^{\text{Neg}} [^{\text{+neg}} [^{\text{KNOW}} \text{Neg}]^{\text{+neg, KNOW}} \text{Neg}] \rightarrow \ [^{\text{Neg}} [^{\text{+neg}} [^{\text{KNOW}} \text{Neg}]^{\text{+neg, KNOW}} \text{Neg}]^{\text{+neg, KNOW}} \text{Neg}]

\((13) \ a. \ [+\text{neg}, \text{KNOW}] \leftrightarrow /\text{molu}/ \quad b. \ [\text{KNOW}] \leftrightarrow /\text{al}/ \quad c. \ [+\text{neg}] \leftrightarrow /\text{an(i)}/

Because (13a) is the most highly specified among the compatible vocabulary items with the features for the fused node, it is chosen to be inserted to this node, resulting in *molu-n-ta.

The fusion operation is sensitive to the syntactico-semantic features of the terminal nodes and the structural relationship (sisterhood) of these nodes. For example, if the node \([^{\text{v}} [^{\text{\text{+caus}}} \text{KNOW}]^{\text{+caus}}, \text{Neg}] [^{\text{+pres}} \text{T}] \text{Neg}] [^{\text{+neg}}]^{\text{KNOW}} \text{Neg} \] \rightarrow \text{realized as *an(i) al-li-n-ta}

This treatment explains the same syntactic and semantic behaviors of the suppletive negations and the usual short-form negation constructions (negator distribution, negative polarity item licensing, and scope ambiguity). Also, this account allows syntactic structure and operations (overt and covert) to be uniform regardless of the predicate chosen in a clause for the syntactic representation and derivation. It also explains why these suppletive negative forms lack the negator. One theoretical consequence is that morphology does not derive syntax, but interprets it. The proposed analysis also copes with the problem of blocking in a lexicalist approach, the problem of blocking between a lexical entry (*molu-*) and a syntactic construction (*an(i) al-*)